(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: 0 360 527 A2

(12) EUROPEAN PATENT APPLICATION

(21) Application number: 89309445.8

(51) Int. Cl.5: G06F 15/80

(22) Date of filing: 18.09.89

(30) Priority: 19.09.88 JP 234545/88
19.09.88 JP 234546/88

(43) Date of publication of application:
28.03.90 Bulletin 90/13

(84) Designated Contracting States:
DE FR GB

(71) Applicant: FUJITSU LIMITED
1015, Kamikodanaka Nakahara-ku
Kawasaki-shi Kanagawa 211(JP)

(72) Inventor: Shindo, Tatsuya
2-8-23, Sugesengoku Tama-ku
Kawasaki-shi Kanagawa 214(JP)
Inventor: Kawamura, Kaoru
Fujigaoka Roiyaru Mansion 305 1-39-2,
Fujigaoka
Midori-ku Yokohama-shi Kanagawa 227(JP)
Inventor: Umeda, Masanobu
157, Shirahatakamicho Kanagawa-ku
Yokohama-shi Kanagawa 221(JP)
Inventor: Shibuya, Toshiyuki
Shatoru Park 203 2650, Yanokuchi
Inagi-shi Tokyo 210(JP)
Inventor: Miwatari, Hideki
1074-115, Shinoharacho Kohoku-ku
Yokohama-shi Kanagawa 222(JP)

(74) Representative: Stebbing, Timothy Charles et al
Haseltine Lake & Co. Hazlitt House 28
Southampton Buildings Chancery Lane
London WC2A 1AT(GB)

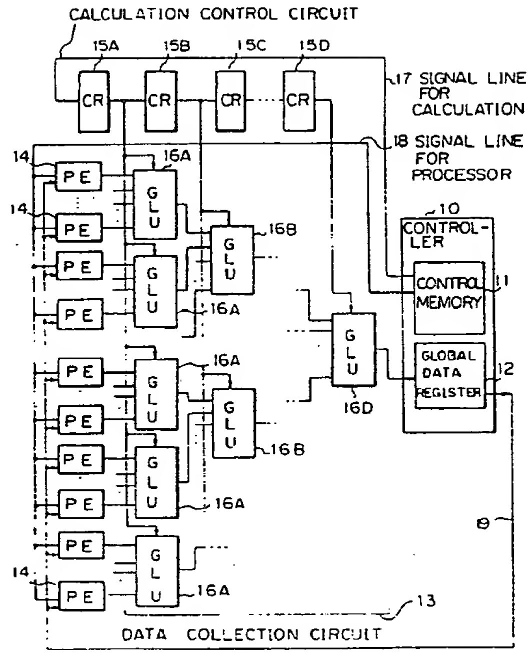
(54) Parallel computer system using a SIMD method.

(57) A parallel computer system using a SIMD method, which in one aspect is constituted by a controller (10) and a plurality of processor elements (14), each of the processor elements has a storage unit to store data to be processed, the controller controls operation of the processor elements, and the parallel computer system performs processing of the data based on a calculation control signal transmitted from the controller (10). The parallel computer system further comprises: a data collection unit (13) connected between the processor elements and the controller (10) for receiving output data from the processor elements (14), performing a predetermined calculation, and outputting calculated data to the controller; and a calculation control unit (15) connected between the data collection unit (13) and the controller (10) for transmitting the calculation control signal from the controller to the data collection unit (13) to make it possible to perform the predetermined calculation in the data collection circuit.

In another aspect (Fig. 10), the parallel computer system comprises: a controller (10), a plurality of control groups (G1...G4), each of the control groups being constituted by a number of processor elements divided from a plurality of the processor elements (14), for use as an address control unit; a plurality of scheduling circuits (110), each of the scheduling circuits provided for one of the control groups (G1...G4) and operatively connected to the controller (10), and for receiving and managing an event signal designating an address signal for data to be processed and transmitted from an adjacent control group; and a plurality of real address generation circuits (120), each of the real address generation circuits provided for one of the control groups (G1...G4) and connected between the controller (10), the scheduling circuit (110), and the control group, for generating an address signal for data to be processed by the processor element (14) belonging to the control group based on a base address determined by the event signal to be managed by the scheduling circuit (110), and an address signal applied from the controller (10).

EP 0 360 527 A2

Fig. 1


CALCULATION CONTROL CIRCUIT
15A
15B
15C
15D
CR
CR
CR
CR
17 SIGNAL LINE FOR CALCULATION
18 SIGNAL LINE FOR PROCESSOR
14
PE
GLU
16A
16B
16D
10
CONTROLLER
CONTROL MEMORY
11
GLOBAL DATA REGISTER
12
19
13
DATA COLLECTION CIRCUIT

PARALLEL COMPUTER SYSTEM USING A SIMD METHOD

The present invention relates to a parallel computer system using a SIMD method constituted by a controller and a plurality of processor elements connected to each other in a lattice configuration.

Parallel computer systems are widely used, particularly, in the field of CAD (Computer Aided Design) which necessitates high speed calculation for a LSI (large scale integrated) circuit design. Accordingly, it is necessary to improve techniques to make these processor elements operate more efficiently in accordance with requirements of high density and high speed LSI.

There are two types of parallel computers based on the connection configuration between the processor elements and the controller. One method is called an MIMD (multiple instruction stream multiple data stream) method which is constituted by a plurality of processor elements and controllers. In this method, each of the processor elements is connected to a corresponding controller, respectively. Accordingly, it is necessary to provide the same number of controllers as there are processors. However, it is difficult to constitute a large scale parallel computer system using this method because a large number of controllers are necessary in accordance with the number of processors, which can be from several tens to several hundreds of processors.

The other method is called an SIMD (single instruction stream multiple data stream) method which is constituted by a plurality of processor elements and one controller. In this method, the controller is connected in parallel to all processor elements. Accordingly, it is possible to constitute a large scale parallel computer which has a large number of processor elements, for example, several tens of thousands of processors.

In the latter method, a "Connection Machine" made by Thinking Machines Corporation uses the SIMD method. This system is constituted by several tens of thousands of processor elements.

However, there are some problems in the SIMD type of parallel computer.

A first problem occurs in the synchronization of all the processor elements. In general, two countermeasures are taken for solving this problem. In one method, data for obtaining synchronization is exchanged between processor elements through a transmission line. However, it is necessary for all the processor elements to be entirely connected to effectively apply this method. In the other method, a particular signal for obtaining synchronization is output from each processor element. Then, a "wired-OR" logic is performed on all of the synchronization signals and the resultant data of the wired-OR is returned to all of the processor elements. However, the number of processor elements is limited in the wired-OR logic approach because a large delay occurs in the wired-OR logic operation.

A second problem occurs in the order of priority use of the bus line. When the number of processor elements reaches from several thousand to several tens of thousands, it is necessary to determine the priority order of use of the bus line.

A third problem occurs in the extraction of essential data from essential processor elements. The essential data is, for example, maximum data or minimum data.

One type of parallel computer system according to the present invention is provided in view of the above problems.

The other type of the parallel computer system according to the present invention can control all processor elements so as to effectively and uniformly distribute the processor elements as a load.

Embodiments of the present invention may provide a parallel computer system using a SIMD method enabling high efficiency data processing and high load distribution capability.

According to one aspect of the invention, there is provided a parallel computer system using a SIMD method constituted by a controller and a plurality of processor elements, each of the processor elements having a storage unit to store data to be processed, the controller controlling the operation of the processor elements, and the parallel computer system processing data based on a calculation control signal transmitted from the controller, the parallel computer system comprising: a data collection unit connected between the processor elements and the controller for receiving data output from the processor elements, performing a predetermined calculation, and outputting calculated data to the controller; and a calculation control unit connected between the data collection unit and the controller for transmitting the calculation control signal from the controller to the data calculation unit to make it possible to perform the predetermined calculation in the data collection circuit.

According to another aspect of the present invention, there is provided a parallel computer system using a SIMD method constituted by a controller and a plurality of processor elements, each of the processor elements having a storage unit to store data to be processed, the controller controlling operation of the processor elements, and the parallel computer system performing processing of data based on a calculation control signal transmitted from the controller, the parallel

computer system comprising: a plurality of control groups, each control group being constituted by a number of processor elements divided from a plurality of processor elements, to be utilized as an address control unit; a plurality of scheduling circuits, with a scheduling circuit being provided for each control group and operatively connected to the controller, for receiving and managing an event signal designating an address signal for data to be processed and transmitted from an adjacent control group; and a plurality of real address generation circuits with a real address generation circuit provided for each control group and operatively connected to the controller, the scheduling circuit and the control group, for generating an address signal for data to be processed by a processor element belonging to the control group based on a base address determined by the event signal to be managed by the scheduling circuit and an address signal applied from the controller.

Reference is made, by way of example, to the accompanying drawings in which:-

Fig. 1 is a basic block diagram of a type of parallel computer system embodying the first aspect of the invention;

Fig. 2 is one embodiment of a parallel computer system shown in Fig. 1;

Fig. 3 is a schematic block of a processor element shown in Figs. 1 and 2;

Fig. 4 is a schematic block diagram of a gathering logic unit GLU shown in Fig. 1;

Fig. 5 is a table for explaining various control signals shown in Fig. 4;

Fig. 6 shows one example of a gathering logic unit shown in Fig. 1;

Fig. 7 is a signal timing chart for explaining the operation of the circuit shown in Fig. 6;

Fig. 8 is a detailed block diagram of a gathering logic unit shown in Fig. 4;

Fig. 9 is detailed block diagram of the MAX/MIN/ADD calculation circuit shown in Fig. 4;

Fig. 10 is basic block diagram of a type of parallel computer system embodying the second aspect of the present invention;

Fig. 11 is a schematic block of a processor element shown in Fig. 10;

Fig. 12 is a view for explaining the concept of the computer system shown in Fig. 10;

Fig. 13 is a view for explaining the division of a virtual area shown in Fig. 12;

Figs. 14A and 14B are views for explaining addresses of memory spaces shown in Fig. 12;

Fig. 15 is a view for explaining control groups shown in Fig. 10;

Fig. 16 is a block diagram of control groups and peripheral circuits;

Fig. 17 is a block diagram for explaining a pseudo processor element;

Fig. 18 is a detailed block diagram of a scheduling circuit shown in Fig. 10;

Fig. 19 is a detailed block diagram of an input circuit for the window number shown in Fig. 18;

Fig. 20 is a detailed block diagram of a consecutiveness detection circuit shown in Fig. 18;

Fig. 21 is a detailed block diagram of an event input circuit shown in Fig. 18;

Fig. 22 is a logic table in an event interpretation circuit shown in Fig. 18;

Fig. 23 is a detailed block diagram of a FIFO circuit shown in Fig. 18;

Fig. 24 is a detailed block diagram of a registration flag circuit shown in Fig. 18;

Figs. 25A to 25C are detailed block diagram of an address calculation circuit shown in Fig. 18; and

Fig. 26 is a detailed block diagram of a real address generation circuit shown in Fig. 18.

Figure 1 is a basic block diagram of one type of parallel computer system embodying the first aspect of the invention. In Fig. 1, reference number 10 denotes a controller, 11 a control memory for storing a micro-code including output control signals, and 12 a global data register for performing an input/output operation of the data processed or to be processed. The control memory 11 and the global data register 12 are provided in the controller 10. Reference number 13 denotes a data collection circuit for collecting output data from processor elements (PE) 14. 15A to 15D denote control registers CR constituting a calculation control circuit and connected to each other using a pipe-line method for applying various calculation control signals to the collection circuit 13. 16A to 16D denote gathering logic units (GLU) constituting the collection circuit 13 and each constituted by a tree configuration. Reference number 17 denotes a signal line for the calculation control signal to the GLU, 18 a signal line for controlling processor elements, and 19 a data line for broadcasting global data.

Each of the processor elements comprises a data register for storing the data to be processed and an arithmetic logic unit ALU as shown in Fig. 3. The arithmetic logic unit ALU calculates the data stored in the register in response to the order transmitted from the controller 10 through the signal line 18.

Each gathering logic unit GLU 16A to 16D collects the output data transmitted from the processor elements. The gathering logic units 16A to 16D are connected in the form of a tree configuration having several stages. That is, in Fig. 1, the units 16A are the first stage, the units 16B are the second stage, and the unit 16D is the final stage. The outputs of the processor elements 14 are input to the gathering logic units 16A. The resultant

calculation data obtained in the GLU's 16A are output to the GLU's 16B. Similarly, the resultant data obtained in the GLU's 16B are output to the next stage. The final stage 16D gathers all resultant data obtained in the previous stages and the data calculated in the final stage 16D is output to the global data register 12 in the controller 10.

Each of the calculation control registers CR 15A to 15D are connected in series with each other using the pipe-line method. The number of registers is equal to the number of stages in the gathering logic unit GLU. In this case, the calculation in each stage is performed in response to the calculation control signals, for example, an "ADD" calculation signal, transmitted through the signal line 17. That is, when the calculation signal "ADD" is input to the first stage 16A, the calculation suggested by the calculation signal is performed in the first stage 16A regarding the data output from the processor elements. This calculation signal is transmitted to the next stage in response to the clock signal from the controller 10 and the same calculation suggested by the calculation signal is performed in the second stage 16B. The above calculation is performed using the pipe-line method. That is, when the first calculation signal "ADD" is input to the first stage, the next calculation signal, for example, "MAX" is input to the first stage.

The synchronization of all processor elements is performed in accordance with a synchronization signal from the controller 10. The controller 10 sends the synchronization signal to all processor elements through the control line 18 to output the value "1" when each processor element completes the predetermined processing. At the same time, the signal "AND" is transmitted to the control register 15A through the control line 17.

When the calculation signal "AND" is set in the register 15A, the GLU 16A of the first stage performs an "AND" calculation regarding all output from the processor element in response to the first clock. The same "AND" calculation is performed in the GLU 16B of the next stage in response to the next clock. When the same "AND" calculation is performed in the GLU 16D of the final stage in response to the clock and the resultant data is the value "1", the controller 10 can recognize that all processor elements output the value "1".

The essential processor element having the essential data is extracted as follows. A proper processor number is previously attached to each processor element. First, the controller 10 commands the essential processor element to output the proper number. Second, the controller 10 commands another processor element to output a suitable signal, for example, the value "11---1" or "00---0". The controller 10 then sends the control signal "MAX" or "MIN" to the control register 15A. Accordingly, the essential processor element can be selected in response to "MAX" or "MIN" of the number in the collection circuit 13. In this case, a next essential processor element can be selected from the remaining processor elements excluding the first essential processor element in the same manner as the above. Accordingly, it is possible to use this circuit to select the priority order of use of a bus line.

Figure 2 illustrates one particular embodiment of the system shown in Fig. 1. The same reference numbers as used above indicate the same components. In Fig. 2, reference number 20 denotes a processor array the elements of which are connected to each other in a lattice configuration. As explained above, the processor array and the collection circuit are controlled by the control signals from the controller 10. The control memory 11 in the controller 10 comprises a plurality of control formats 1 to n.

The controller 10 further comprises a sequencer 21 which determines the sequence for reading out the control information from the control memory 11. The global data register 12 is a register for holding the data transmitted in common to all processor elements and to receive the output data from the collection circuit 13.

Figure 3 is a schematic block of a processor element. In Fig. 3, reference number 30 denotes a data register for holding the data to be processed. 31 denotes an arithmetic logic unit ALU for calculating the data stored in the register 30. The processor element 14 is controlled by the same control signal transmitted from the controller 10. This control signal includes an address of the data register 30 and an operation code for the arithmetic logic unit 31. The processor element 14 further comprises four ports, i.e., east port (E), west port (W), north port (N) and south port (S) for communicating between adjacent processor elements. The processor element 14 further comprises an input terminal GT for inputting the data from the global data register 12, and a collection terminal CT for outputting the data.

The processor element 14 is a one-bit type and the input/output operation to the data register 30 is basically performed for each bit. Data larger than one bit is processed from the most significant bit (MSB) or the least significant bit (LSB) for each bit.

Figure 4 is a schematic block diagram of a gathering logic unit GLU. Each of the gathering logic units GLU 16A to 16D comprises an OR calculation circuit 40, an AND calculation circuit 41, a MAX/MIN/ADD calculation circuit 42 and a selector circuit SEL 43. The signal GLI is data input to the GLU 16 and constituted by 32 bits. This means that one GLU 16 can handle a maximum of 32 bits

in the input side. The signal GLO is resultant data and is constituted by one bit.

Kinds of calculation control signals input to the GLU 16 from the control registers 15A to 15D are as follows. These control signals are shown in a table in Fig. 5.

XGOPS (2 bits)

This is an operation code for the GLU 16. When this signal is the value "00", the AND calculation is performed, when it is "01", the OR calculation is performed, when it is "10", the "MIN" or "MAX" calculation is performed, and when it is "11", the "ADD"calculation is performed.

XGCR (1 bit)

This is a carry control signal for a carry clear operation in the "ADD" calculation.

GLSTS (2 bits)

This is a switching signal for switching the number of bits of the input signal to the GLU. That is the number of bits of the input signal, for example, 32 bits, 16 bits, 8 bits, or 4 bits can be selected by this switching signal.

GMAXS (1 bit)

This is an instruction signal to select either a "MIN" or "MAX" calculation when the operation order XGOPS is the value "10".

GNOPS (1 bit)

This is an instruction signal to force the input signal to "0". When this bit is "0", the input data GLI becomes invalid.

Figure 6 shows one example of a gathering logic unit GLU. As is obvious from the drawing, the first stage is constituted by four GLU's 16-1 to 16-4, and the second stage (final stage) is constituted by one GLU 16-5. Since each GLU can handle 32 bits in the input side, this circuit can handle up to 128 bits in the input side.

The operation code XGOPS and the carry control signal XGCR are used in the registers 15-1 and 15-2 to simplify the explanation. Accordingly, the GLU performs the calculation under the above operation code and the carry control signal. Further, reference numbers 50 to 58 denote registers for the pipe-line control. OP-1 denotes a signal set in the register 15-1, and OP-2 a signal set in the register 15-2. Further, D1 denotes data set in the registers 50 to 53, D2 data set in the registers 54 to 57, and D3 data set in the register 58.

Figure 7 is a signal timing chart for explaining the operation of the circuit shown in Fig. 6. In Fig. 7, B0 to B3 denote four-bit data to be processed, B0 is a least significant bit (LSB) and B3 is a most significant bit (MSB). In the "ADD" operation, these bits are sequentially input to the registers 50 to 53 from the LSB to the MSB, bit by bit, in response to a clock signal CLK. Synchronizing with the data, the operation code "ADD" for addition is set in the register 15-1. The carry control signal XGCR is input to the GLU so as to indicate the carry clear with the value "0" in the first clock, and so as to indicate the normal calculation with the value "1" in the next clock.

The first stage GLU 16-1 to 16-4 performs an addition calculation for 32 bits regarding the bit B0 (LSB) under the order OP-1 in response to the first clock. The first stage 16-1 to 16-4 next performs an addition calculation of 32 bits regarding the bit B1 in response to the next clock. In this addition calculation, the carry operation is taken into consideration for the resultant data of the bit B0. These calculations are continued until the bit B3.

The resultant data obtained by the GLU 16-1 to 16-4 is set in the register 54 to 57. In this case, the order OP-1 is moved to the OP-2 at every clock.

The final stage GLU 16-5 performs the addition calculation for four input signals in response to the order OP-2. The resultant data is output to the register 58. Since the collection circuit is constituted by two stages, the GLU 16-5 performs the same operation as that of the GLU 16-1 to 16-4 after a delay of one clock.

As explained above, the calculation is performed using the pipe-line method. Regarding the calculation of "MAX" and "MIN", the same calculation as mentioned above can be performed with an optional bit length. In this case, the bits are input from the MSB to the LSB.

Figure 8 is a detailed block diagram of a gathering logic unit shown in Fig. 4. In Fig. 8, A1 denotes an AND circuit, N1 to N5 denote NOR circuits, NA-1 to NA-5 denote NAND circuits, S1 to S5 denote selectors, and MA00 to MA40 denote MAX/MIN/ADD calculation circuits. The AND circuit A1 sets the input data GLI to all zeroes when the control signal GNOPS is the value "0". The NOR circuits N1 to N4 and the NAND circuit perform the OR calculation in accordance with the number of input stages regarding the input data GLI. The NAND circuits NA2 to NA5 and the NOR circuit N5 perform the AND calculation regarding the input data GLI. The MAX/MIN/ADD calculation circuits

MA00 to MA40 perform the maximum/minimum/addition calculation. The selectors S1 to S4 select the output in accordance with the operation code XGOPS. The selector S5 performs the selection in accordance with the number of the input stage.

Figure 9 is detailed block diagram of the MAX/MIN/ADD calculation circuit. In Fig. 9, A10 to A16 denote AND circuits, N10 to N12 NOR circuits, NT0 to NT6 NOT circuits, 01 to 02 OR circuits, R0 to R1 registers, S10 to S11 selectors, and 90 a full adder having three inputs.

This circuit performs a maximum/minimum calculation for 2 inputs. DM0 and DM1 are the input signals each having one bit to obtain a maximum/minimum value. DA0 and DA1 are the input signals each having one bit for addition. XM is resultant data for a maximum/minimum calculation and XA is resultant data for addition. In the first stage MA00 to MA15, the DM0 is equal to the DA0, and the DM1 is equal to the DA1, respectively.

The operation of the addition "ADD" is explained in detail below. The addition data is input to the DA0 and DA1 from the LSB bit at every bit. In the first bit, since the signal XGCR is the value "0", the carry is cleared so that the carry CARRY-0 becomes "0". The adder 90 performs the addition regarding the DA0 and DA1, and the resultant data XA is output therefrom. When no carry is performed in the addition, the output signal CARRYOUT becomes "0". When a carry is performed, the output signal CARRYOUT becomes "1". The signal CARRYOUT is held in the register R0 for use in the addition at the next clock through the selector S10. In the next bit from the LSB, the content of the register R0 is used as the carry CARRY-0, and is added to the DA0 and DA1.

The operation for obtaining the maximum value is explained in detail below. When obtaining the maximum value, the signal GMAXS is the value "0", the input data is input to the DM0 and DM1 at every bit from the MSB bit. In the first bit, the signal XGCR is "0", and the outputs of the AND circuit A10 and A11 are "0" so that the output of the NOR circuit N12 becomes "1" and the outputs of the AND circuits A14, A15 and the OR circuit 01 become "0". Accordingly, the selection signal in the selector S11 becomes "10" so that, as the maximum output XM, the output of the OR circuit 02 is selected in the OR logic operation between the DM0 and DM1. In the registers R0 and R1, when one of the DM0 and DM1 previously becomes "1", the value "1" is set to the corresponding side. That is, when the DM0 is "1" and the DM1 is "0", the value "1" is set to the register R0. On the contrary, when the DM0 is "0" and the DM1 is "1", the value "1" is set to the register R1.

When the value "1" is input to one of the registers R0 and R1, the output of the NOR circuit N12 becomes "0" from the next clock. Further, the output of the OR circuit 01 becomes "1" when the register R0 is "1" and becomes "0" when the register R1 is "1". Accordingly, after the above selection, the selector 11 outputs the value of either the DM0 or the DM1 in which the value "1" was previously detected.

The operation for obtaining the minimum value is explained in detail below. When obtaining the minimum value, the signal GMAXS is the value "1". The operation is the same as that for the maximum. When the output of the NOR circuit N12 is the value "1", the selection signal to the selector S11 is "11" and the output of the AND circuit A16 is selected. When one of the DM0 and DM1 becomes "1", one of the registers R0 and R1 becomes "1" and the selection signal to the selector S11 becomes "00" or "01". After the above selection, the minimum value in either DM0 or DM1 is selected.

Figure 10 is a basic block diagram showing a type of parallel computer system embodying the second aspect of the present invention. In Fig. 10, reference number 110 denotes a scheduling circuit SC, 120 a real address generation circuit RAGC, and 150 a pseudo processor element. Further, G1 to G4 denote control groups to be used as a control unit for accessing the address. Accordingly processor elements are divided into several control groups. The scheduling circuit 110 and the real address generation circuit 120 are provided for each control group.

The scheduling circuit 110 is a circuit for receiving an event signal to designate the address and for managing the address designated by the event signal by using a queue.

The real address generation circuit 120 is a circuit for generating a real address of the data to be processed by the processor element belonging to that control group. This generation is performed based on a base address determined by the event signal and an address signal applied from the controller 10.

The pseudo processor element 150 is provided in the boundary portion of each control group. The pseudo processor element 150 has a function of sending the data corresponding to the address of the processor element when the processor element located to the boundary portion gives and takes the data between the adjacent processor elements belonging to the adjacent control group. This circuit is provided to ensure consecutiveness between the processor elements.

Figure 11 is a schematic block diagram of a processor element shown in Fig. 10. This drawing is the same as Figure 3 except that an external memory 200 is added between the data register 30

and the real address generation circuit 120. The address of the external memory 200 is applied from the real address generation circuit 120 provided in every control group. This type of parallel computer system embodying the second aspect of the invention mainly related to the address control for the external memory 200.

Figure 12 is a view for explaining the concept of the type of system shown in Fig. 10. Reference number 301 denotes an actual processor element group, 302 a first memory space corresponding to the actual processor element group 301, and 300 a second memory space (virtual area) corresponding to a virtual processor element group. Accordingly, the first memory space 302 coincides with an object area to be processed by the actual processor element group 301. In general, the object area to be processed (for example, a wire pattern area) coincides with the size of the actual processor element group. However, in this type of system the object area can be widened up to the second memory space. In this case, the actual processor element group 301 moves to the second memory space 300 so that it is possible to process data regarding the larger object area exceeding the first memory space. Therefore, although the virtual processor element group does not actually exist it is possible to obtain the same performance as the processor element group having the second memory space 300 by moving the actual processor element group 301.

Figure 13 is a view for explaining division of the virtual area shown in Fig. 12. The second memory space 300 is divided into a plurality of windows (m x n window). Accordingly, one window corresponds to the first memory space 302 processed by the actual processor element group 301. The window number is attached to each window from 0 to nm-1, respectively.

Figures 14A and 14B are views for explaining addresses of memory spaces. In Fig. 14A, the external memory 200 of one processor element 14 is divided into sixteen memory spaces for the virtual processor element. That is, "0000" to "FFFF" are addresses for the external memory each having sixteen bits, while "000" to "FFF" are addresses for the virtual area each having twelve bits. Accordingly, one actual processor element functions as sixteen virtual processor elements.

In Fig. 14B, the window number denotes the base address indicating the head of each memory space of the virtual PE (processor element) and constituted by eight bits "aaaa 0000" since a maximum of 256 windows can be provided. Since the external memory 200 is divided into sixteen blocks in this embodiment, the lower four bits are set to "0000". The virtual PE address "0000 bbbbbbbbbbbb" denotes the relative address of each memory space of the virtual PE. The virtual PE address is transmitted in common to all processor elements from the controller 10. The virtual PE address has "0000" in the upper bits in accordance with the number of the window. As explained in Fig. 14A, when the number of the window is sixteen, the virtual PE address is constituted by twelve bits. As shown in Fig. 14B, the real address "aaaabbbbbbbbbbbb" having sixteen bits of the external memory 200 can be obtained by adding (or performing an OR operation) the base address and the virtual PE address.

The processing of the data in the virtual PE is performed in such a way that the real PE sequentially processes the corresponding data in the virtual memory space divided from the real external memory 200. In this case, as the simplest method, there is a method in which the real PE always sequentially processes all virtual PE's including its own external memory. However, this method is not efficient because the virtual PE's in which the processing is not necessary are included. Accordingly, this aspect of the present invention selects the virtual PE's in which the processing is necessary so that the efficiency of the processing can be raised. Therefore, the concept of the "event" is employed to realize this method in this aspect of the invention.

The event is started when the conditions to be processed to the virtual PE are realized. The virtual PE which received the event is handled as the object to be processed by the real PE. The controller determines the content of the event in accordance with a program.

Figure 15 is a view for explaining control groups shown in Fig. 10. As shown in the drawing, the processor elements (PE) 14 are divided into the control groups G1, G2, ---. For example, the PE's of 128 x 128 are divided into sixteen control groups G1 to G16 each having 32 x 32 PE's.

Figure 16 is a block diagram of control groups and peripheral circuits. In Fig. 16, G1 to G16 are control groups, 110 (SC) is the scheduling circuit provided for each control group, and 120 is a real address generation circuit also provided for each control group. The scheduling circuit 110 receives the event from the PE and manages the virtual PE to be processed. The virtual PE number to be processed, i.e., the window number, is queued in the scheduling circuit 110 and sequentially processed from the head of the queue. The scheduling circuit 110 sends the base address corresponding to the virtual PE to the real address generation circuit 120. Accordingly, the scheduling circuit 110 performs the queueing and assigns the real PE.

The real address generation circuit 120 generates the real address based on the relative address of the virtual PE and the base address. In

this case, the relative address indicates a kind of control signal transmitted in common from the controller to all PE's, and the base address is determined by the scheduling circuit 110. The real address is transmitted to the real PE's in each control group.

The scheduling circuit 110 is connected to four adjacent scheduling circuits. Each input/output signal is explained below.

Event signal (as input signal)

This event signal is obtained by the OR logic among the event signals transmitted from all PE's (32 PE's in this embodiment) located on the boundary of the control group, and is used as the input signal. This signal is one bit for four directions of E, W, N, and S.

Window number signal (as input signal)

The window number signal of the adjacent scheduling circuit 110 is input as the input signal. The window number signal has eight bits as shown in Fig. 14B for four directions of E, W, N, and S. The scheduling circuit inputs the corresponding window number to the event signal when that event signal is activated, and performs the queuing.

Self-event signal (as input signal)

This signal is obtained by the OR logic among all event signals of the PE's included in its own control group, and has one bit.

Window number signal (as output signal)

This signal is the window number signal output to the adjacent scheduling circuit 110, and has eight bits for four directions of E, W, N, and S.

Base address signal (as output signal)

This signal is an output signal to the real address generation circuit 120 indicating the corresponding address to the window number of the virtual PE read-out from the head of the queue.

Various control signals (as input/output signal)

These signals are output or input signals to or from the controller 10. For example, the control signal NEXT is a signal to indicator reading out a next virtual PE from the queue, and the control signal DIR is a signal to indicate the direction of the data flow in four directions E, W, N, and S. The control signal EMPTY is a signal to indicate vacancy of the input signal, the clock signal, and the queue.

Figure 17 is a block diagram for explaining the pseudo processor element (PE) shown in Fig. 10. In Fig. 17, the boundary BD of the control group is provided between the processor elements 14A and 14B. That is, the PE 14A is adjacent to the PE 14B. The pseudo PE (PS-PE) 150A is provided adjacent to the PE 14A, and the pseudo PE 150B is provided adjacent to the PE 14B, respectively.

The pseudo PE is provided for ensuring the consecutiveness of the processing between adjacent control groups. This is because the adjacent control group can not receive the necessary value of the window when the object window between the adjacent control groups is different. Accordingly, as shown in Figs. 10 and 17, the pseudo PE is provided to each end of the row of the PE's in each control group. Therefore, when the object window is consecutive between adjacent control groups, the pseudo PE's are not used and the PE 14A directly accesses the PE 14B by switching the selectors S1 and S2.

When the PE 14A performs the read/write (R/W) operation to its own external memory 200A, the write data is simultaneously written to the external memory 200a belonging to the pseudo PE 150A. When the PE 14A transmits the data to the PE 14B, the pseudo PE 150A read the data from the external memory 200a and transmits that data to the PE 14B through the selector S1 instead of the PE 14A. The address of the external memory 200a is the window address of the PE 14B side. The same operation as the above is performed in case of the data transmission from the PE 14B to the PE 14A. Although this drawing shows the connection of one direction as the lattice of one dimension, it is possible to connect two directions as a lattice of two dimensions.

Figure 18 is a detailed block diagram of the scheduling circuit shown in Fig. 10 and Figures 19 to 25 are detailed circuits of the diagram in Fig. 18. In Fig. 18, reference number 500 denotes an input circuit for the window number, 510 a registration table, 520 a consecutiveness detection circuit, 530 an input circuit for the event, 540 an event interpretation circuit, 550 a first-in/first-out (FIFO) circuit, 560 a registration flag circuit, 570 an address holding circuit, and 580 an address calculation circuit. Further, R1 to R4 denote registers for the pipe-line control.

The input circuit 500 inputs the window number determined from the four adjacent directions E, W,

N, and S, where DIR is the control signal for indicating the data flow. This circuit is shown in detail in Fig. 19.

In Fig. 19, R10 denotes a register for holding the window numbers input from four directions E, W, N and S. S10 denotes a selector for selecting the window number in response to the control signal DIR and outputting the selected window number having eight bits.

The registration table 510 is a table for storing flags indicating whether or not the window number is registered. One bit is assigned to each window in a maximum of 256 windows. Accordingly, the window number from the input circuit 500 becomes the address in the table 510. Therefore, double registration of a window number is prevented by this method.

The consecutiveness detection circuit 520 determines the consecutiveness between the present area and the adjacent area. The detailed circuit is shown in Fig. 20.

In Fig. 20, COMP denotes a comparator, 600 an encoder (ECD), OR an OR circuit, and S20 a selector (SEL). CE, CW, CN and CS denote registers for storing the resultant data of the detection of the consecutiveness until the reset signal is input. The comparator COMP compares the upper bits of the address of its own control group with the window number input from the input circuit 500. When the former coincides with the latter, the encoder 600 outputs an enable signal in response to the direction control signal DIR. The enable signal is stored in the registers CE, CW, CN and CS as consecutiveness data and the consecutiveness data C-FLAG is output from the selector S20 in response to the control signal DIR.

The event input circuit 530 receives the event signals from four directions. The detailed circuit is shown in Fig. 21.

In Fig. 21, EVCLR denotes an event clear signal to clear each register R. S30 denotes a selector circuit. The register R is cleared by the event clear signal EVCLR. When the event signal is loaded in the register R, the event signal is output from the selector S30 through the AND circuit.

The event interpretation circuit 540 judges whether or not the queuing of the window number should be performed, or whether or not the present address should be held. The detailed logic table for determining the output from this circuit 540 is shown in Fig. 22.

In Fig. 22, T denotes an active state of the signals. The registration signal REG indicating the queuing of the window number is output from the circuit 540 only when the output of the input circuit 530 is active. The address holding signal AHS is output when the consecutiveness signal C-FLAG and the event signal are active. Further, the address holding circuit is output when the self-event signal is active.

The FIFO 550 stores the window number to be processed in accordance with the event signal. The detailed circuit is shown in Fig. 23.

In Fig. 23, MEM denotes a memory having the capacity of 8 x 256 bits, R40 to R43 registers, S40 a selector, WCNT a write counter to output the write address, RCNT a read counter to output the read address, and COMP a comparator. When the registration signal is set in the register R41, the window number stored in the register R40 is written to the address indicated by the write counter WCNT in the memory MEM. Further, the content of the address of the memory MEM is read out in response to the control signal NEXT through the register R42 and the AND circuit, and output through the register R43. When the comparator detects coincidence between the content of the write counter WCNT and the content of the read counter RCNT, a signal EMPTY indicating the vacant state is output.

The registration flag circuit 560 is shown in detail in Fig. 24. In Fig. 24, 700 denotes an encoder, and R a register. The direction of the registered window number is stored in the register R after being encoded by the encoder 700 in accordance with the direction control signal DIR.

The address calculation circuit 580 outputs the window number to be informed to the adjacent control group and the upper address bits used for generation of the real address based on the window number read out from the FIFO 550. The detailed circuit is shown in Figs. 25A to 25C.

In Fig. 25A, in the boundary of the window, the control group sends the window numbers (A + 1) and (A - 1) for the horizontal direction, and sends the window numbers (A + B) and (A - B) for the normal direction, where B denotes the number of the window for the transverse direction when the virtual area is divided into the plural windows.

In Fig. 25B, the boundary of the window is distinguished by the boundary marks (E, W, N, S) 80. The value of each boundary mark is set by the controller 10 in the initial stage.

In Fig. 25C, ALU denotes calculation circuit, R80 to R82 registers, and S80 to S82 selectors. The calculation circuit ALU calculates any of the window numbers A, A ± 1, and A ± B in accordance with the boundary mark E, W, N, S shown in Fig. 25B. An address designation value ADD-DEG indicates a mode using the address transmitted from the controller 10 as an absolute address regardless of the present window number. When this mode is designated, the address designation value ADD-DEG is transmitted to the real address generation circuit 120 through the selectors S80 and S82.

Figure 26 is a detailed block diagram of the real address generation circuit. In Fig. 26, R100 to R105 denote registers, S100 to 103 selectors, and OR denotes OR circuits. The input signals to this circuit are the relative address of the virtual PE transmitted from the controller 10, the upper address bits output from the address calculation circuit 580, and the adjacent window numbers input from the input circuit 500. The real address to the PE belonging to its own control group is generated by adding the relative address set in the register R100 to the upper address bits set in the register R101 as shown in Fig. 14B. As shown in Fig. 14B, in the upper eight bits, when the base address and the relative address overlap, one side is set to "0". The real address is obtained by the logic OR calculation. In this case, the lower eight bits of the real address are the same bits as transmitted from the controller 10.

Further, to generate the real address for the adjacent pseudo PE, the window number of the adjacent PE is set in the registers R102 to R105. Further, the window numbers are controlled by the selectors S100 to S103 to be the address of the adjacent control group when loading (L), and to be the self-address when saving (S).

In this embodiment, although the multi-processor is constituted by lattice coupling, it is possible to constitute it by hyper-cubic coupling in accordance with the application.

Claims

1. A parallel computer system using a SIMD method constituted by a controller and a plurality of processor elements, each of said processor elements having a storage means to store data to be processed, said controller controlling operation of said processor elements, and said parallel computer system performing processing of said data based on a calculation control signal transmitted from said controller, said parallel computer system comprising:
a data collection means connected between said processor elements and said controller for receiving output data from said processor elements, performing a predetermined calculation, and outputting calculated data to said controller, and
a calculation control means connected between said data collection means and said controller for transmitting said calculation control signal from said controller to said data calculation means to make it possible to perform said predetermined calculation in said data collection circuit.

2. A parallel computer system as claimed in claim 1, wherein said data collection means comprises a plurality of gathering logic units coupled to a tree configuration having a plurality of stages, a first stage of said gathering logic unit receiving said output data from each of said processor elements, a second stage of said gathering logic unit receiving calculation data obtained from said first stage, calculation data obtained from said second stage being output to a next stage, and final calculation data obtained from a final stage being output to said controller.

3. A parallel computer system as claimed in claim 1, or 2, wherein said calculation control means comprises a plurality of control registers corresponding to a number of said stage, each of the control registers being connected by a pipe-line method, and each of the control registers sequentially outputting said calculation control signal to each of said stages.

4. A parallel computer system as claimed in claim 1 or 3, wherein said calculation control signal comprises an AND calculation signal, an OR calculation signal, a MAX/MIN calculation signal, and an ADD calculation signal.

5. A parallel computer system as claimed in claim 1, 2, or 4, wherein said gathering logic unit comprises an OR calculation circuit for performing a logic OR calculation of data output from each of said processor elements, an AND calculation circuit for performing a logic AND calculation of data output from each of said processor elements, a MAX/MIN/ADD calculation circuit for obtaining a maximum value, minimum value, or added value of data output from each of said processor elements, and a selector for selecting said OR calculation circuit, AND calculation circuit, and MAX/MIN/ADD calculation circuit, these calculation being performed in response to said calculation control signal.

6. A parallel computer system using a SIMD method constituted by a controller and a plurality of processor elements, each of said processor elements having a storage means to store data to be processed, said controller controlling operation of said processor elements, and said parallel computer system performing processing of said data based on a calculation control signal transmitted from said controller, said parallel computer system comprising:
a plurality of control group, each of said control groups being constituted by a number of processor elements divided from a plurality of said processor elements, for use as an address control unit,
a plurality of scheduling means, each of said scheduling means provided for one of said control groups and operatively connected to said controller, and for receiving and managing an event signal designating an address signal for data to be processed and transmitted from an adjacent control group, and

a plurality of real address generation means, each of said real address generation means provided for one of said control groups and connected between said controller, said scheduling means, and said control group, for generating an address signal for data to be processed by a processor element belonging to said control group based on a base address determined by said event signal to be managed by said scheduling means and an address signal applied from said controller.

7. A parallel computer system as claimed in claim 6, wherein each of said control groups further comprises a plurality of pseudo processor elements, each pseudo processor element being connected to each processor element located in a boundary of an adjacent control group, and for transmitting data at data address to be handled by said processor element to the processor element belonging to said adjacent control group.

8. A parallel computer system as claimed in claim 6 or 7, wherein each said processor element and said pseudo processor element further comprise external memories to store read and write data, said write data being simultaneously written into said external memory of said pseudo processor element when said processor element writes data into its own external memory.

9. A parallel computer system as claimed in claim 8, wherein memory space of said external memory of said processor element is divided into a plurality of memory spaces (windows) of virtual processor elements.

10. A parallel computer system as claimed in claim 9, wherein each of said windows comprises a window number as a base address, said virtual processor element comprises a virtual processor element address as a relative address, and a real address of said processor element is obtained by adding said base address to said relative address or by performing a logical OR operation between said base address and said relative address.

Fig. 1

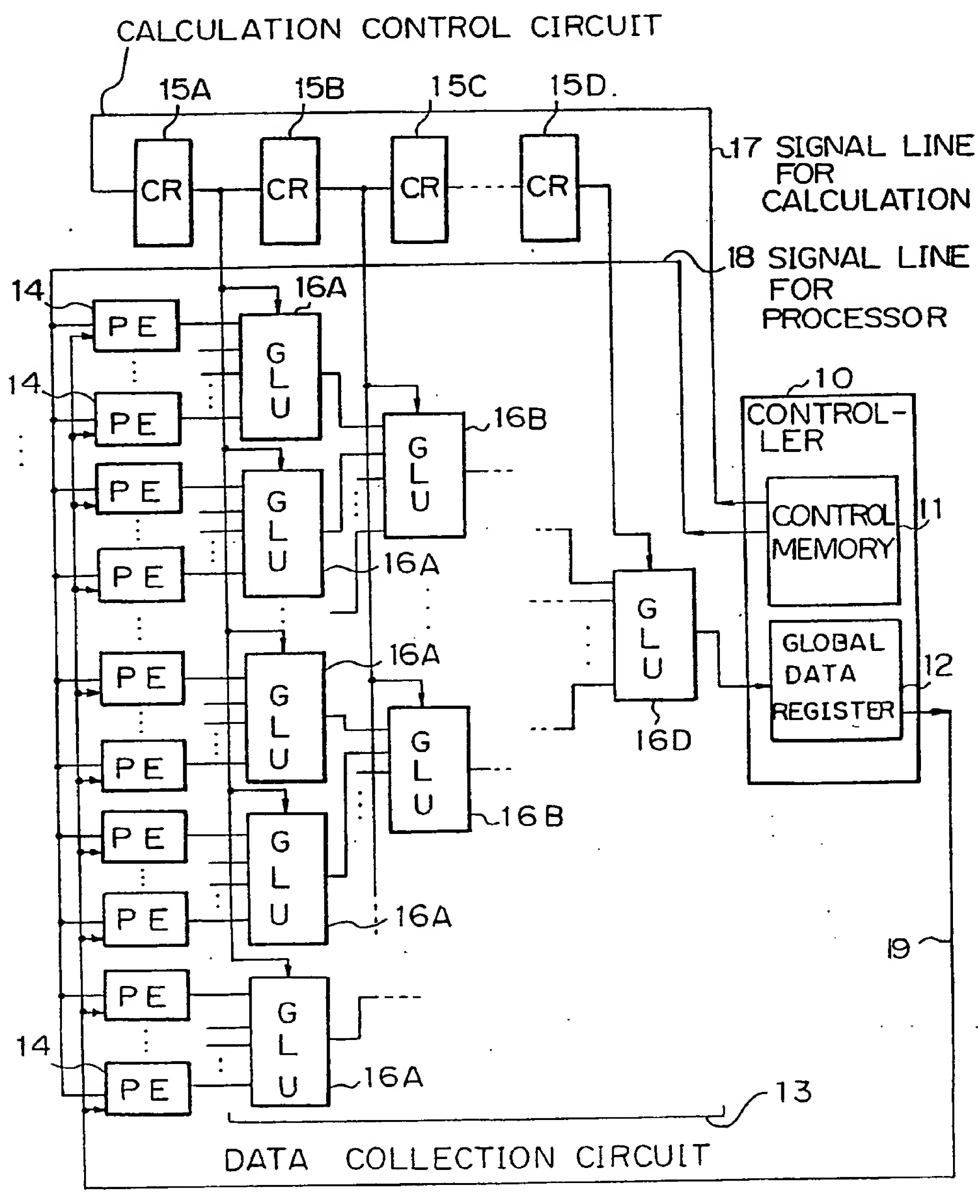

CALCULATION CONTROL CIRCUIT
15A
15B
15C
15D
CR
CR
CR
CR
17 SIGNAL LINE FOR CALCULATION
18 SIGNAL LINE FOR PROCESSOR
14
PE
GLU
16A
16B
16D
10
CONTROLLER
CONTROL MEMORY
11
GLOBAL DATA REGISTER
12
19
13
DATA COLLECTION CIRCUIT

Fig. 2

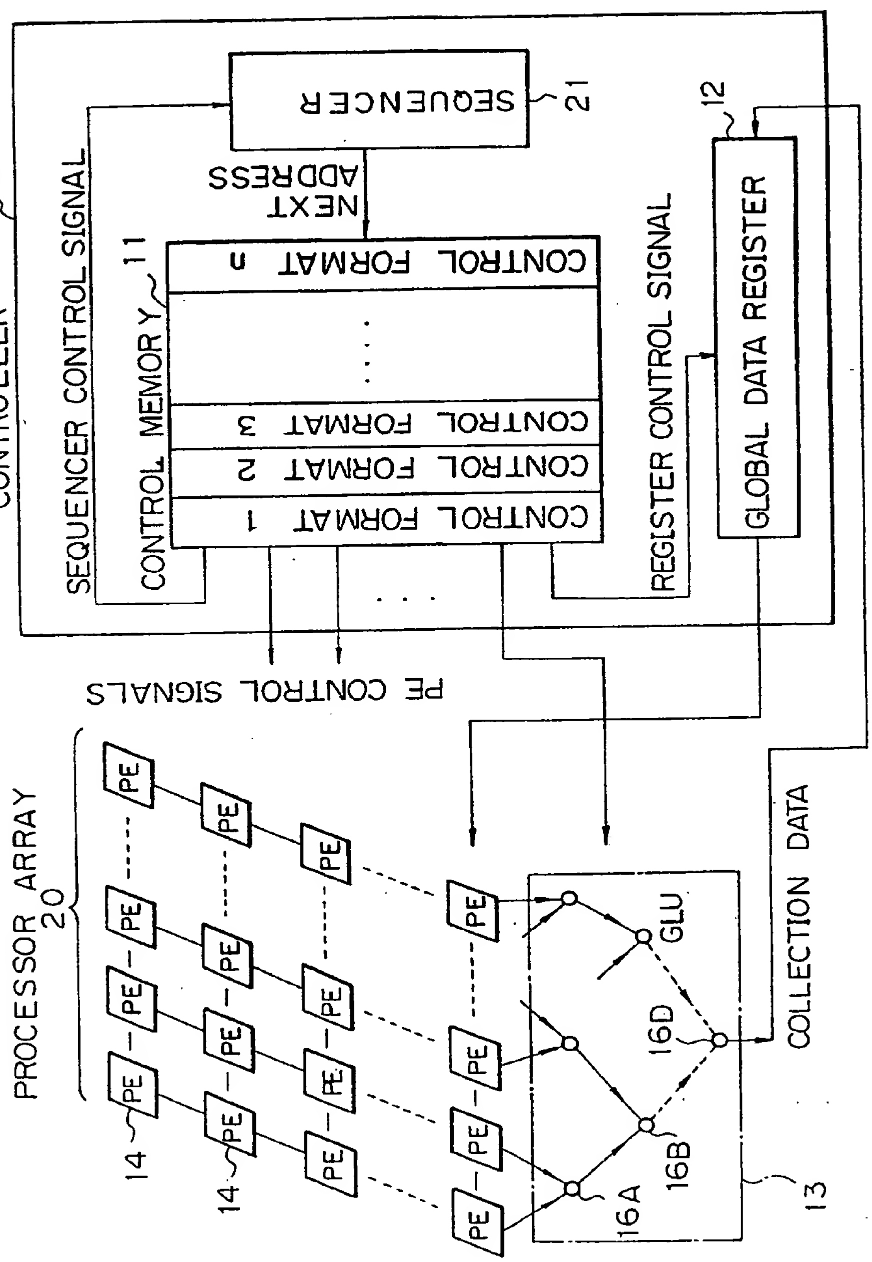

CONTROLLER
10
SEQUENCER
21
NEXT ADDRESS
SEQUENCER CONTROL SIGNAL
CONTROL MEMORY
11
CONTROL FORMAT n
CONTROL FORMAT 3
CONTROL FORMAT 2
CONTROL FORMAT 1
REGISTER CONTROL SIGNAL
GLOBAL DATA REGISTER
12
PE CONTROL SIGNALS
PROCESSOR ARRAY
20
PE
14
GLU
16A
16B
16D
13
COLLECTION DATA

Fig. 3

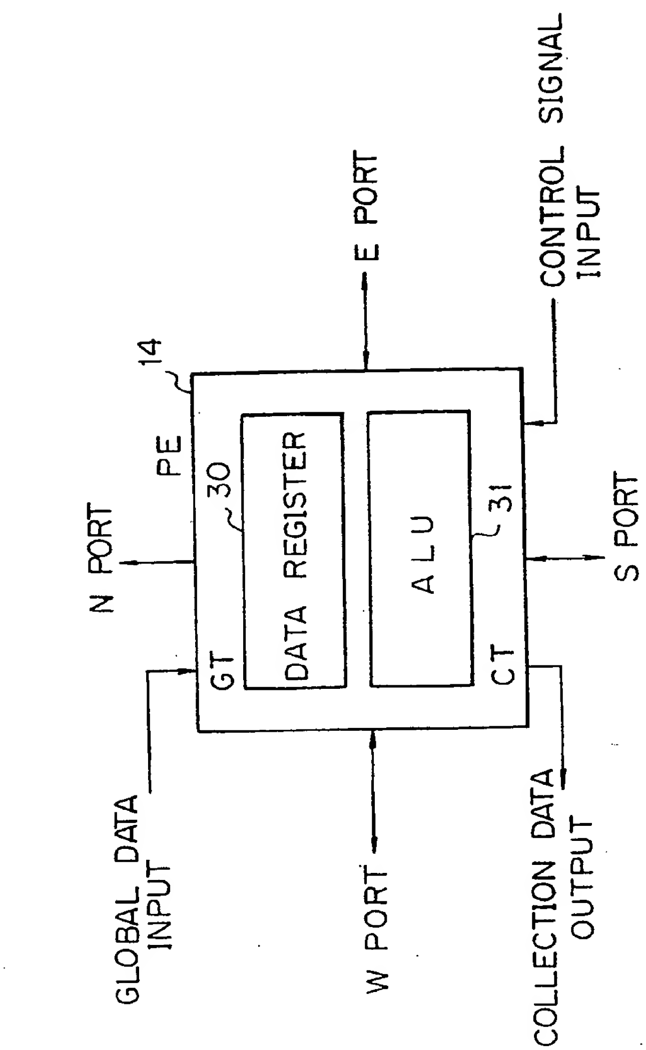

N PORT
PE
14
30
DATA REGISTER
GT
ALU
31
CT
E PORT
CONTROL SIGNAL INPUT
S PORT
GLOBAL DATA INPUT
W PORT
COLLECTION DATA OUTPUT

Fig. 4

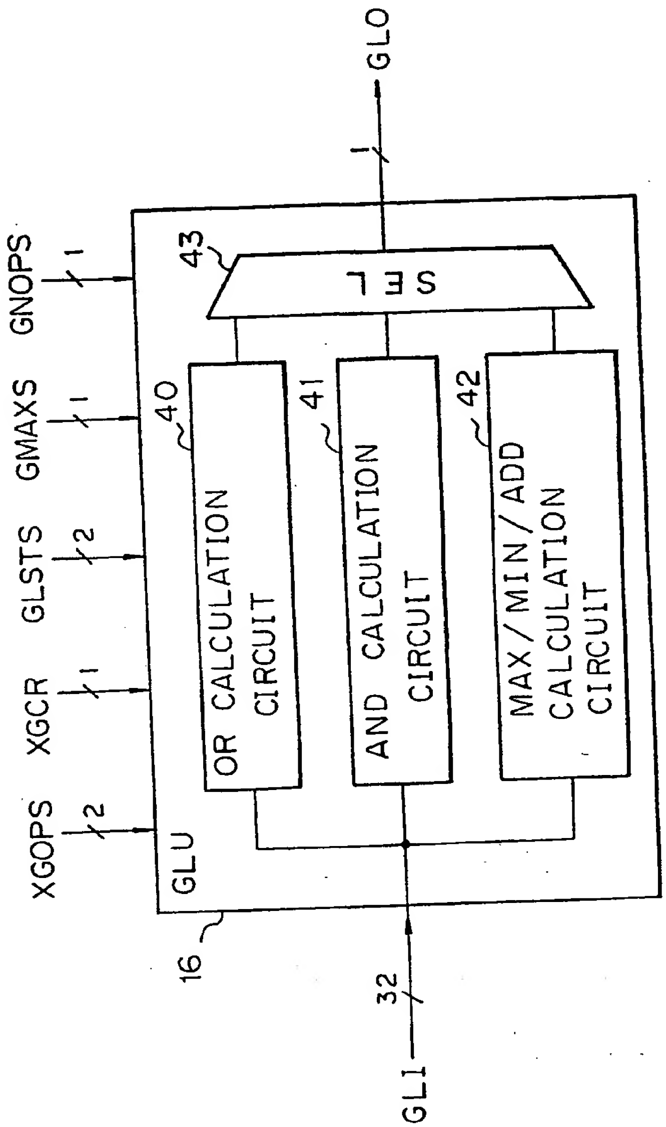

XGOPS
XGCR
GLSTS
GMAXS
GNOPS
GLU
16
GLI
32
OR CALCULATION CIRCUIT
40
AND CALCULATION CIRCUIT
41
MAX / MIN / ADD CALCULATION CIRCUIT
42
SEL
43
GLO
2
1
2
1
1
1

Fig. 5

	SIGNAL NAME	bit	ATTRIBUTE	CONTENTS
DATA	GLI GLO	32 1	I O	GLU INPUT DATA GLU OUTPUT DATA
	TOTAL	33		
CONTROL	XGOPS	2	I	GLU OPERATION CODE (00:AND, 01:OR, 10:MIN/MAX, 11:ADD)
	XGCR	1	I	CARRY CLEAR (0:CARRY CLEAR, 1:NORMAL)
	GLSTS	2	I	GLU INPUT STAGE SW (00:32, 01:16, 10:8, 11:4)
	GMAXS	1	I	MIN, MAX SELECTION SIGNAL (0:MAX, 1:MIN)
	GNOPS	1	I	"0" FORCELY INPUT SIGNAL (0:"0" INPUT, 1:NORMAL)
	TOTAL	7		

TOTAL 40 bits

Fig. 6

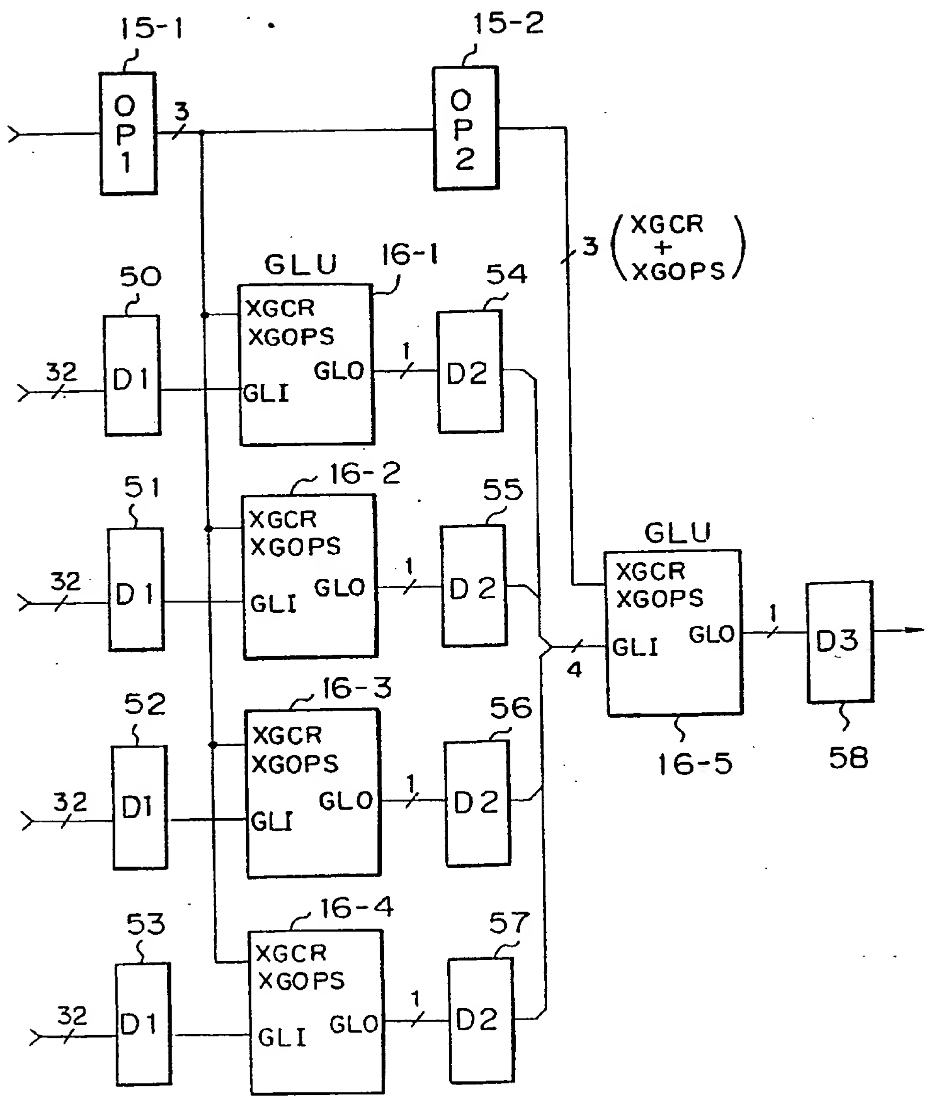
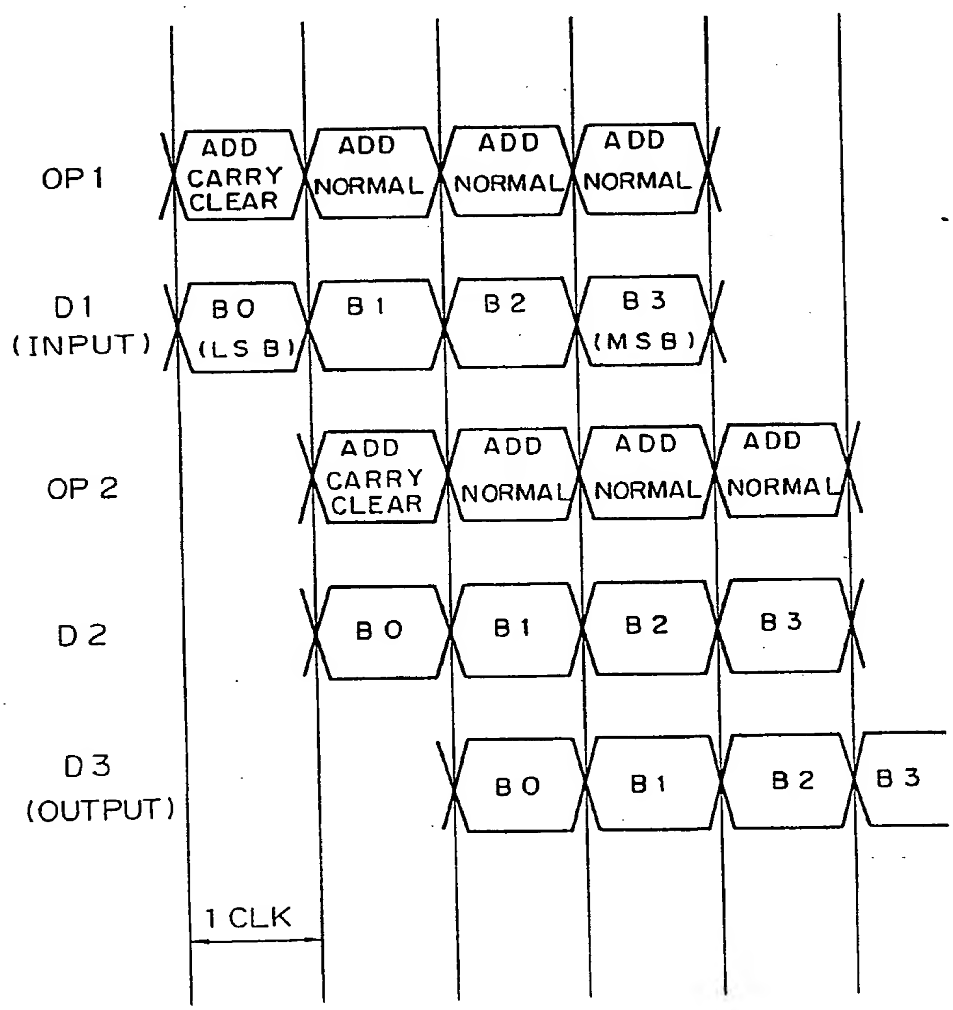

15-1
OP1
3
15-2
OP2
3
XGCR + XGOPS
GLU
16-1
50
32
D1
XGCR
XGOPS
GLI
GLO
1
D2
54
16-2
51
32
D1
XGCR
XGOPS
GLI
GLO
1
D2
55
16-3
52
32
D1
XGCR
XGOPS
GLI
GLO
1
D2
56
16-4
53
32
D1
XGCR
XGOPS
GLI
GLO
1
D2
57
4
GLU
XGCR
XGOPS
GLI
GLO
16-5
1
D3
58

Fig. 7


OP 1
ADD CARRY CLEAR
ADD NORMAL
ADD NORMAL
ADD NORMAL
D1 (INPUT)
B0 (LSB)
B1
B2
B3 (MSB)
OP 2
ADD CARRY CLEAR
ADD NORMAL
ADD NORMAL
ADD NORMAL
D2
B0
B1
B2
B3
D3 (OUTPUT)
B0
B1
B2
B3
1 CLK

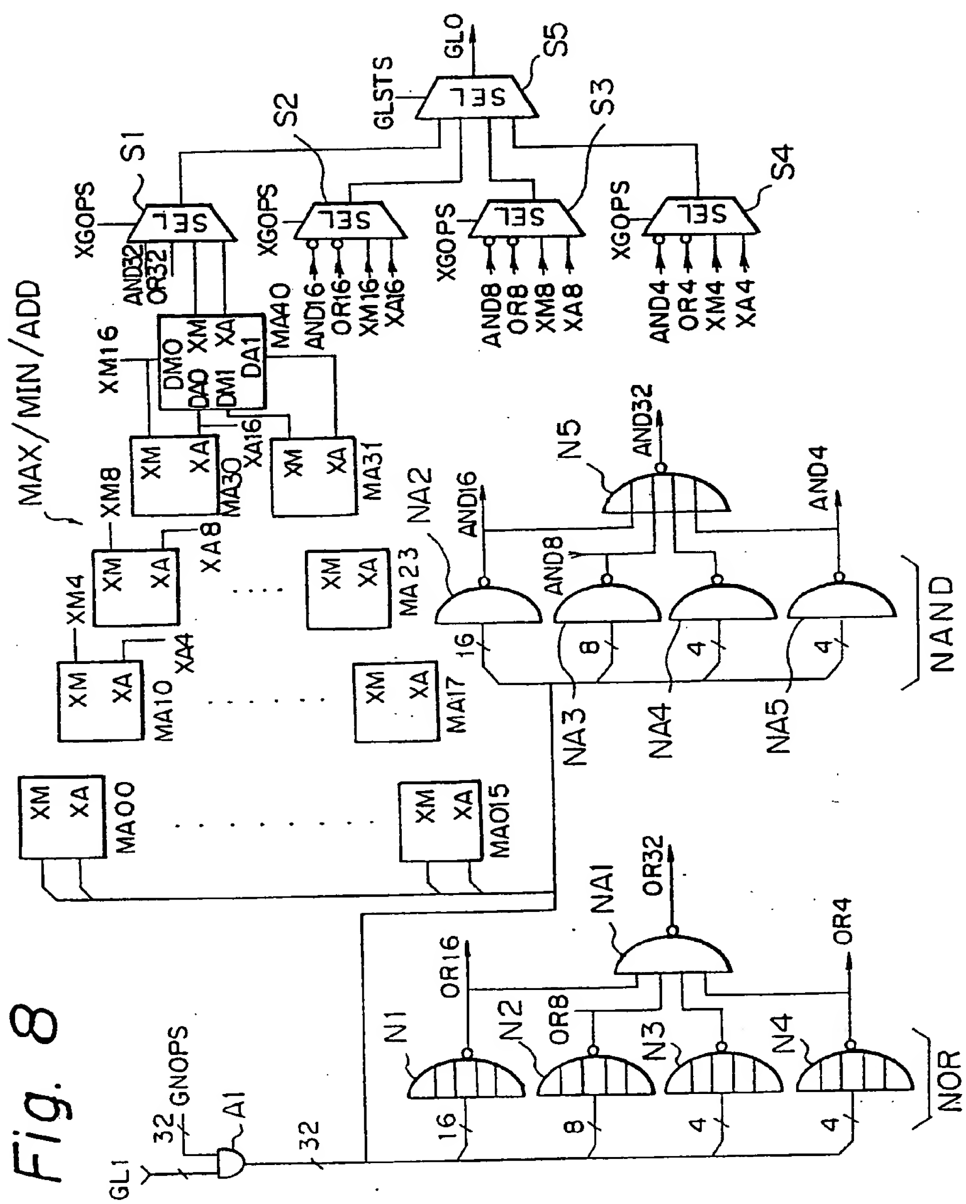

MAX/MIN/ADD
GL1
32
GNOPS
A1
32
XM XA MA00
XM XA MA015
XM XA MA10
XM XA MA17
XM4 XA4
XM XA
XM XA MA23
XM8 XA8
XM XA MA30
XM XA MA31
XM16 XA16
DM0 DA0 DM1 DA1 XM XA
MA40
AND32 OR32
XGOPS
SEL
S1
S2
S3
S4
S5
GLSTS
GLO
AND16 OR16 XM16 XA16
AND8 OR8 XM8 XA8
AND4 OR4 XM4 XA4
NA1
NA2
NA3
NA4
NA5
N1
N2
N3
N4
N5
16
8
4
4
OR16
OR8
OR32
OR4
AND16
AND8
AND32
AND4
NAND
NOR

Fig. 8

Fig. 9

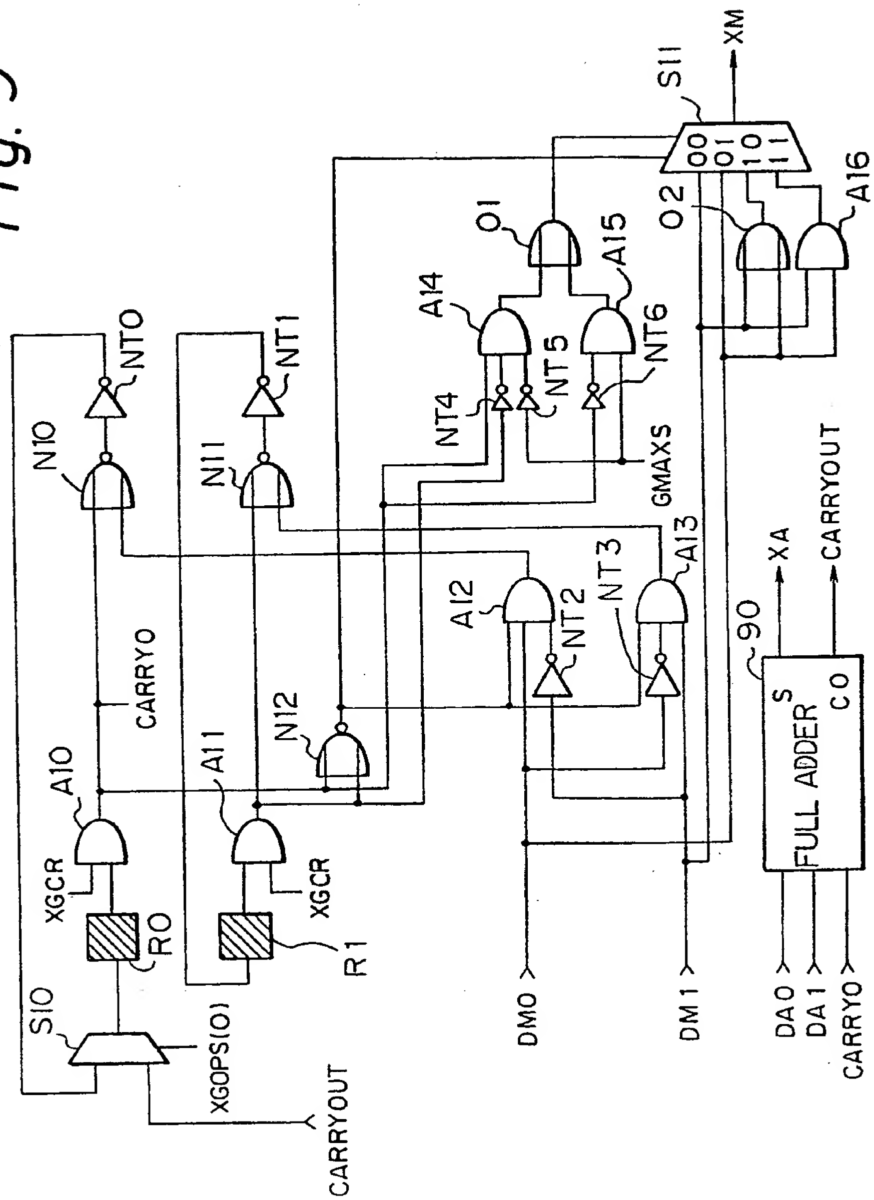

NT0
NT1
N10
N11
A14
NT4
NT5
NT6
O1
A15
S11
XM
00
01
10
11
O2
A16
GMAXS
A12
NT2
NT3
A13
90
XA
CARRYOUT
FULL ADDER
S
CO
CARRY0
N12
A10
A11
XGCR
XGCR
R0
R1
S10
XGOPS(0)
CARRYOUT
DM0
DM1
DA0
DA1
CARRY0

Fig. 10

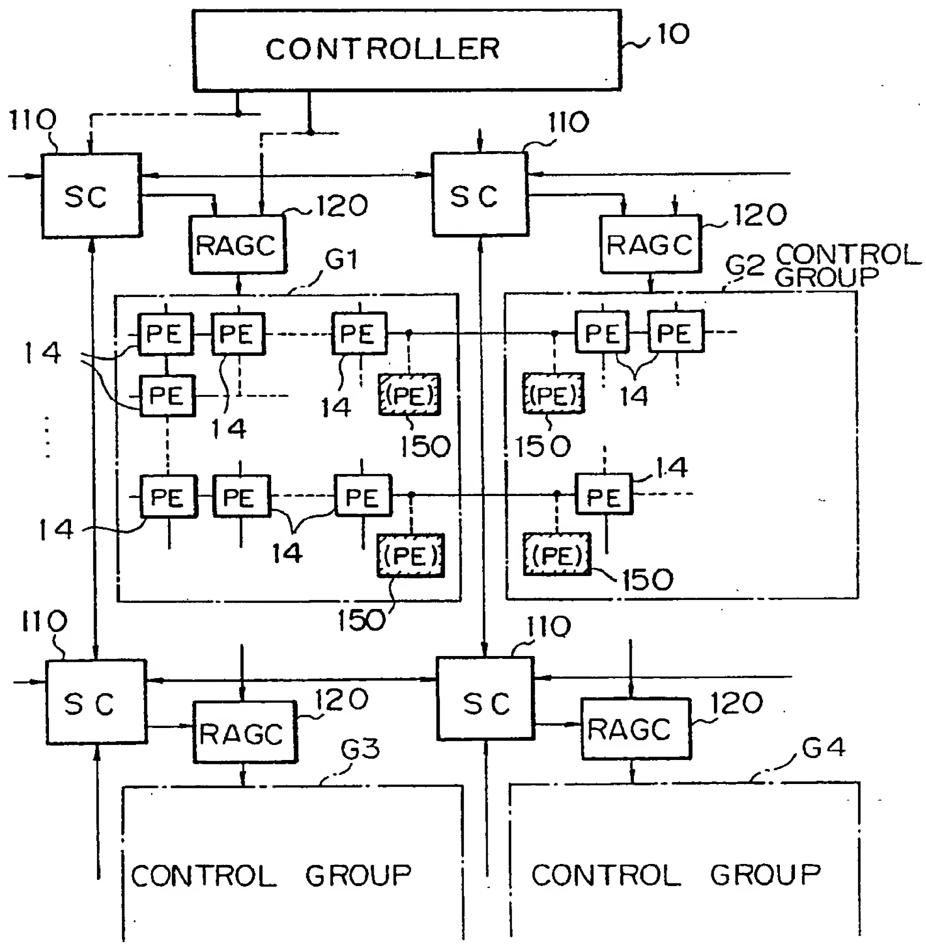

CONTROLLER
10
110
SC
120
RAGC
G1
110
SC
120
RAGC
G2 CONTROL GROUP
PE
(PE)
14
150
110
SC
120
RAGC
G3
CONTROL GROUP
110
SC
120
RAGC
G4
CONTROL GROUP

Fig. 11

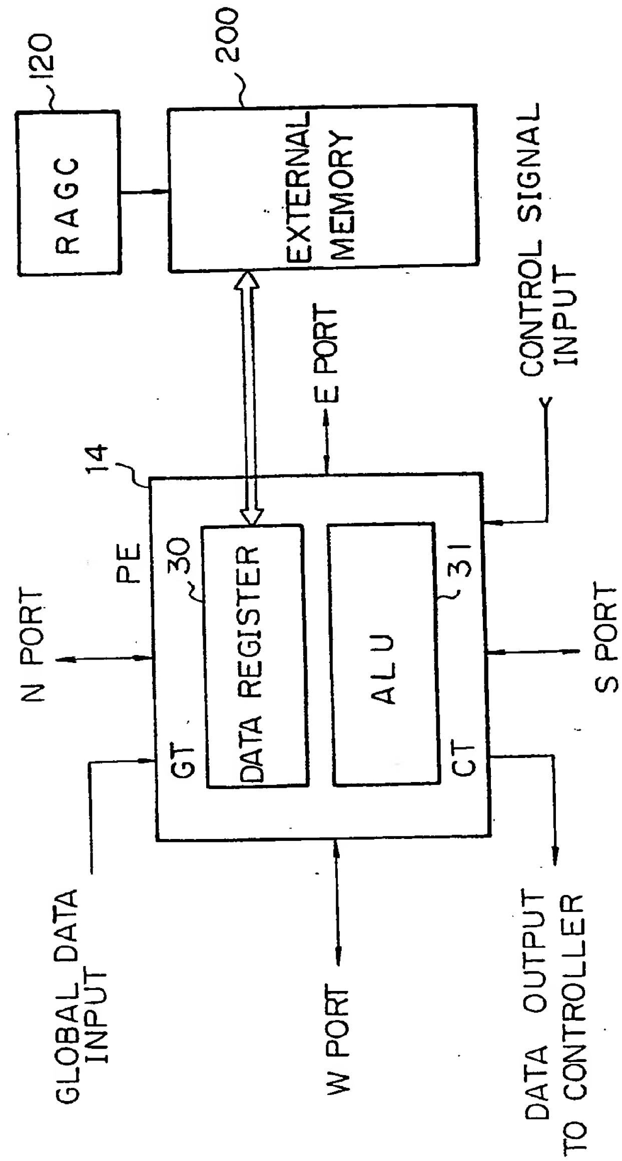

120
RAGC
200
EXTERNAL MEMORY
E PORT
CONTROL SIGNAL INPUT
14
PE
30
DATA REGISTER
31
ALU
GT
CT
N PORT
S PORT
GLOBAL DATA INPUT
W PORT
DATA OUTPUT TO CONTROLLER

Fig. 12

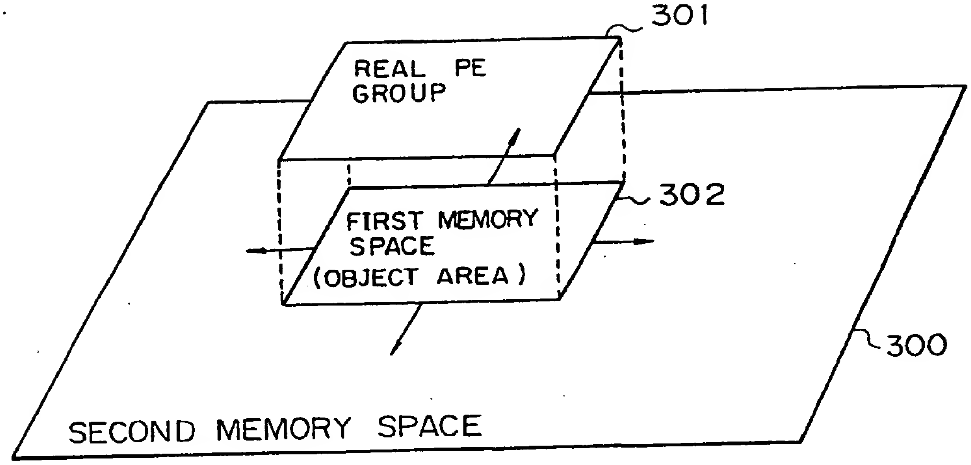

301
REAL PE
GROUP
302
FIRST MEMORY
SPACE
(OBJECT AREA)
300
SECOND MEMORY SPACE

Fig. 13

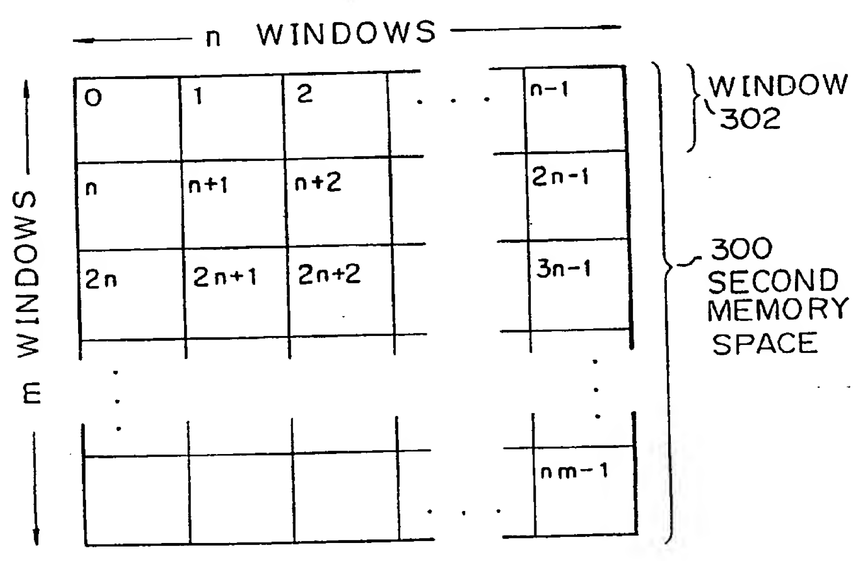

n WINDOWS
m WINDOWS
0
1
2
n-1
n
n+1
n+2
2n-1
2n
2n+1
2n+2
3n-1
nm-1
WINDOW
302
300
SECOND
MEMORY
SPACE

Fig. 14A

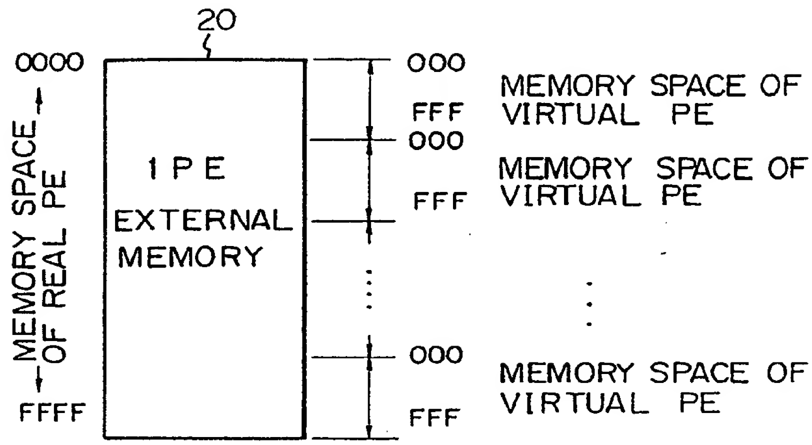


Fig. 14 B

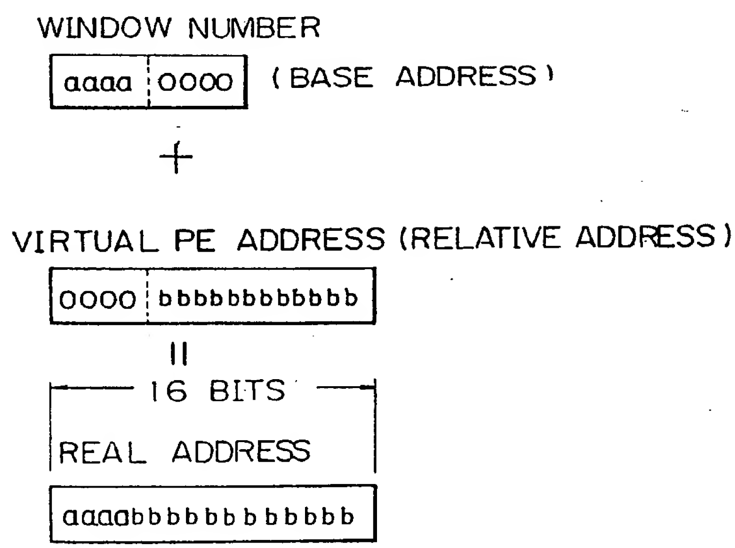


Fig. 15

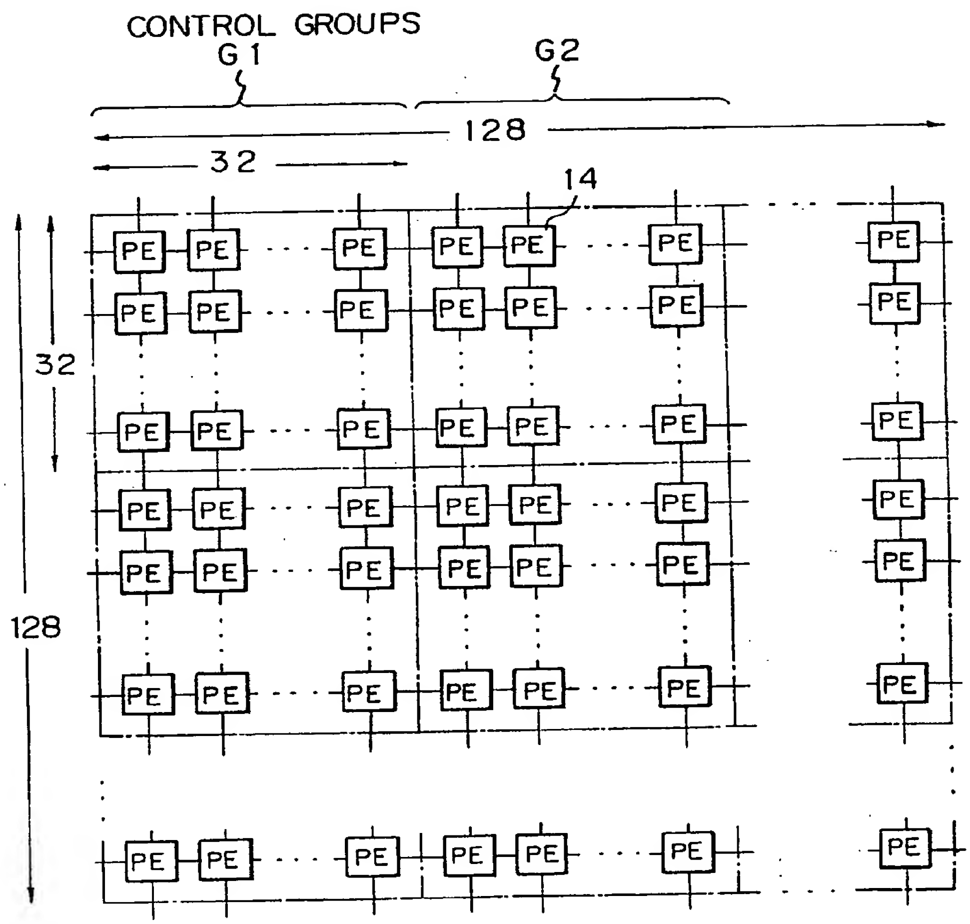


Fig. 16

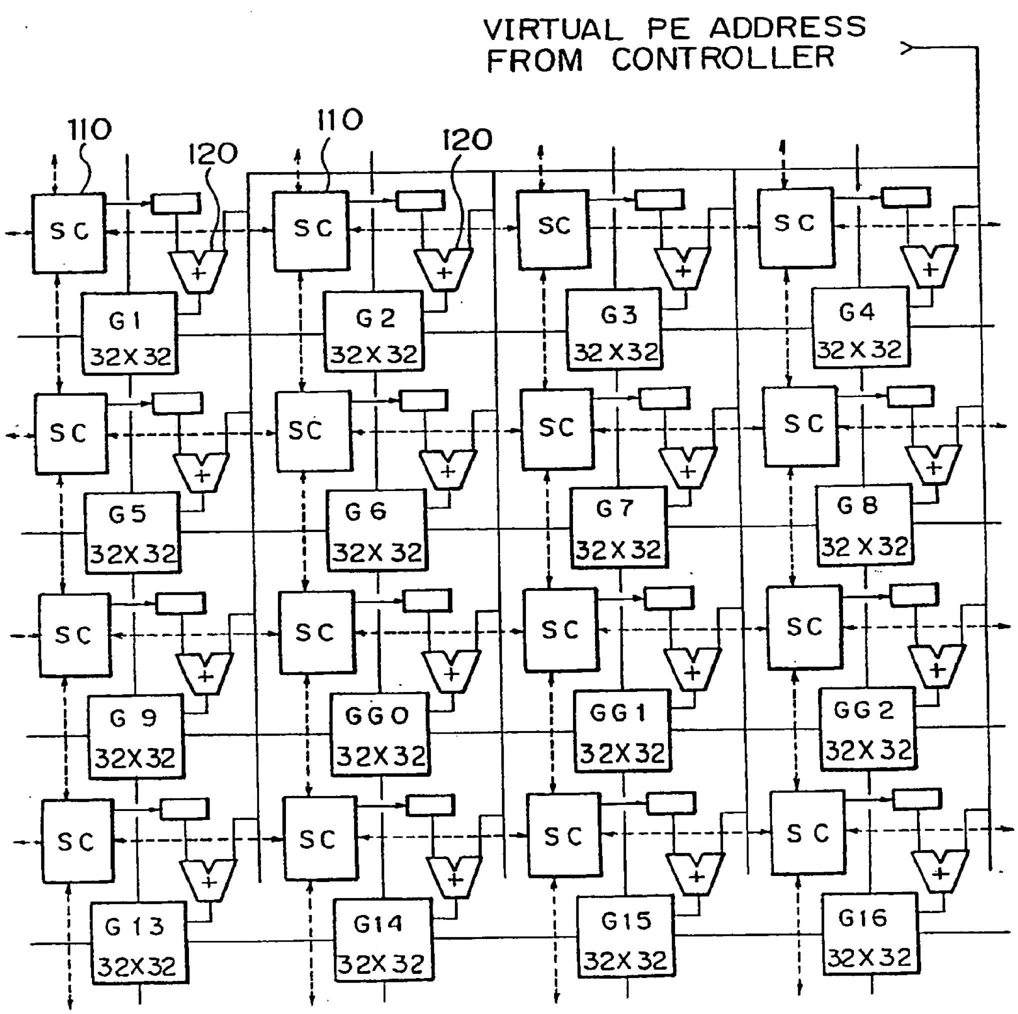

VIRTUAL PE ADDRESS
FROM CONTROLLER
110
120
110
120
SC
G1
32X32
G2
32X32
G3
32X32
G4
32X32
G5
32X32
G6
32X32
G7
32X32
G8
32X32
G9
32X32
GG0
32X32
GG1
32X32
GG2
32X32
G13
32X32
G14
32X32
G15
32X32
G16
32X32

Fig. 17

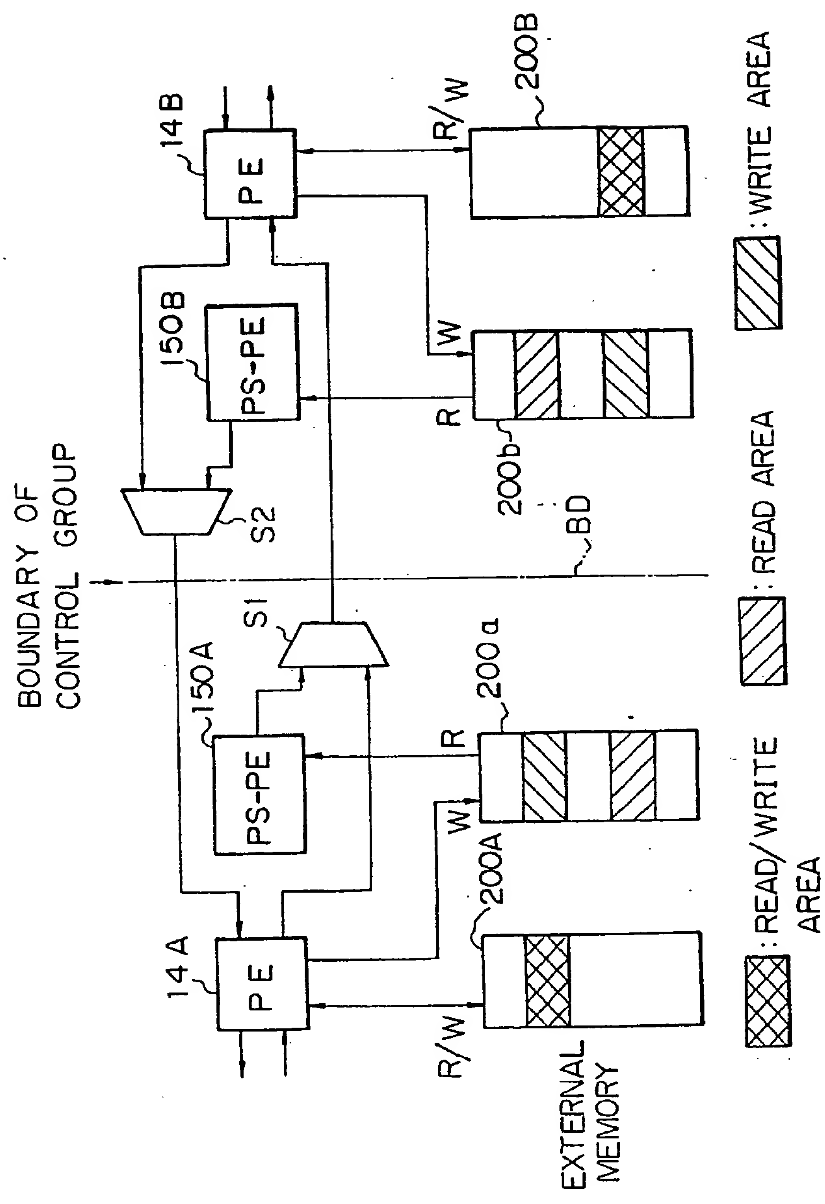

BOUNDARY OF CONTROL GROUP
14B
PE
R/W
200B
150B
PS-PE
W
R
200b
S2
BD
S1
150A
PS-PE
200a
R
W
200A
14A
PE
R/W
EXTERNAL MEMORY
: WRITE AREA
: READ AREA
: READ/WRITE AREA

Fig. 18

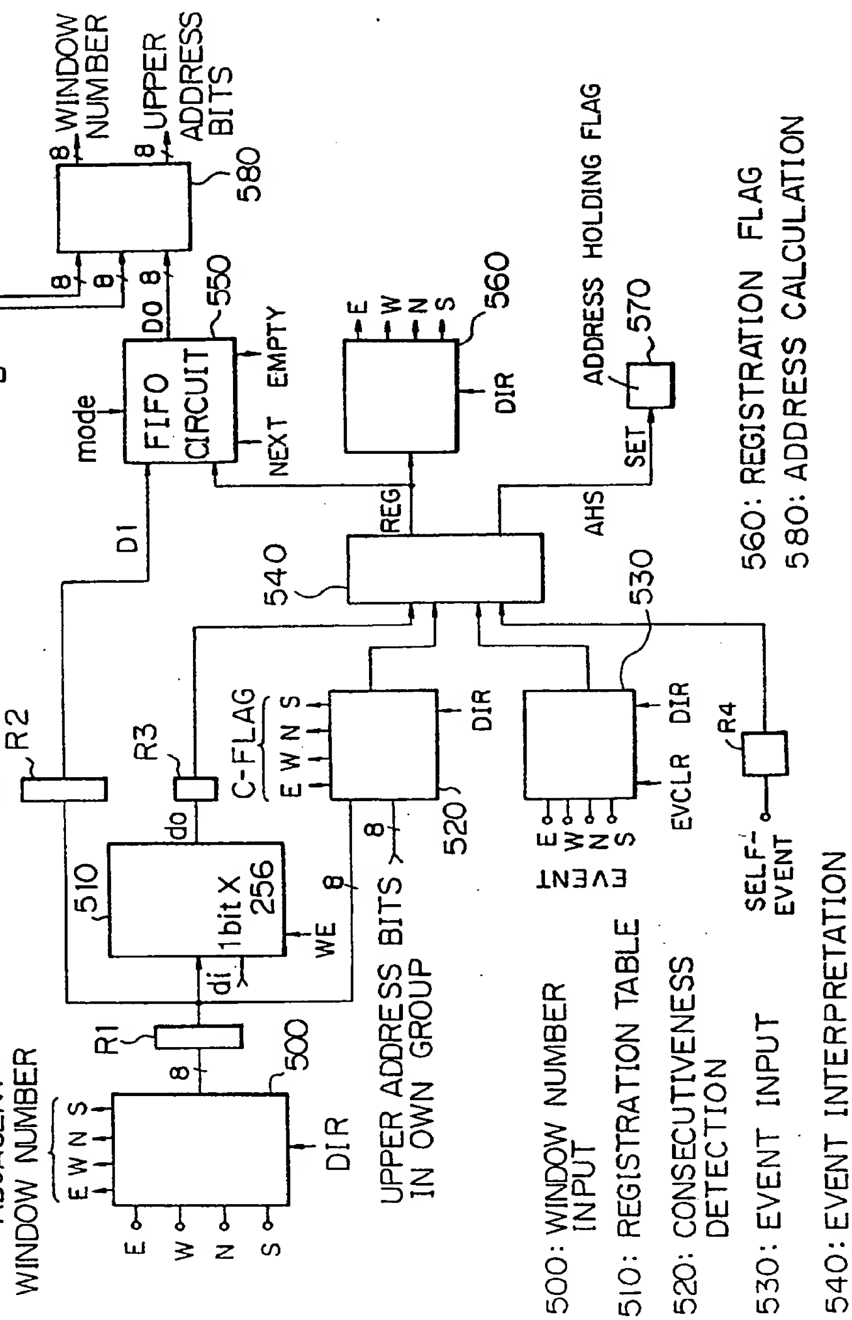

ADJACENT WINDOW NUMBER
E W N S
E W N S
DIR
500
8
R1
di
510
1bit X 256
WE
R2
R3
do
8
C-FLAG
E W N S
520
DIR
UPPER ADDRESS BITS IN OWN GROUP
8
EVENT
E W N S
EVCLR
DIR
530
R4
SELF-EVENT
540
REG
AHS
D1
mode
FIFO CIRCUIT
NEXT
EMPTY
DO 8
550
ADD-DEG
B
8
8
580
8 WINDOW NUMBER
8 UPPER ADDRESS BITS
560
E W N S
DIR
SET
ADDRESS HOLDING FLAG
570
500: WINDOW NUMBER INPUT
510: REGISTRATION TABLE
520: CONSECUTIVENESS DETECTION
530: EVENT INPUT
540: EVENT INTERPRETATION
560: REGISTRATION FLAG
580: ADDRESS CALCULATION

Fig. 19

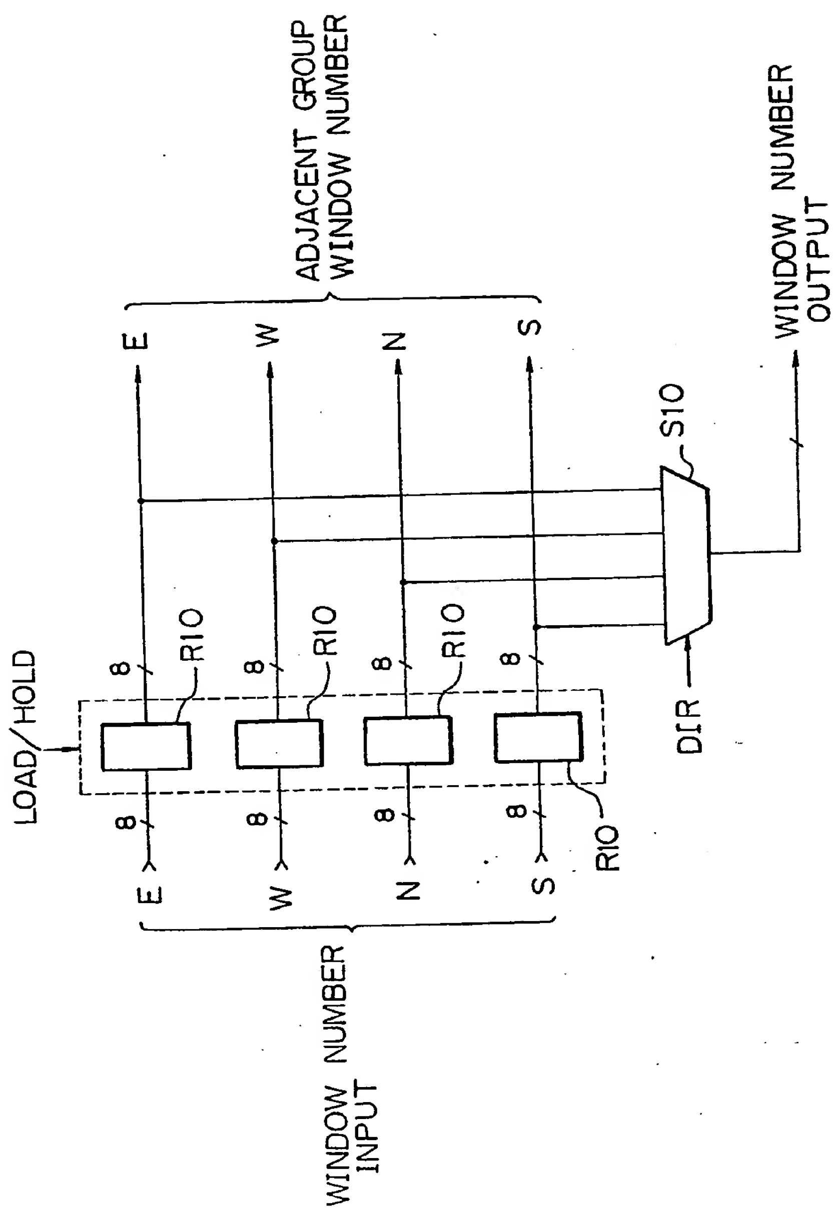

ADJACENT GROUP WINDOW NUMBER
WINDOW NUMBER OUTPUT
WINDOW NUMBER INPUT
LOAD/HOLD
E
W
N
S
8
R10
S10
DIR

Fig. 20

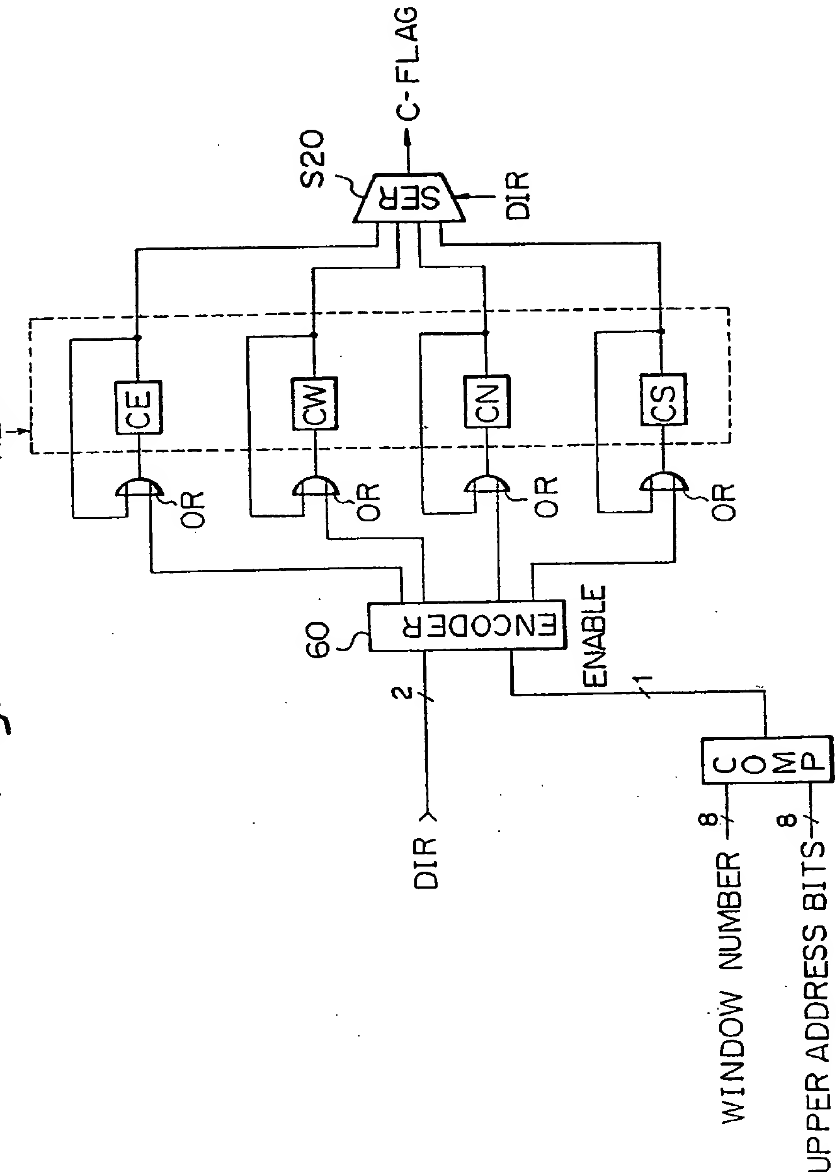

C-FLAG
S20
SER
DIR
RESET
CE
CW
CN
CS
OR
OR
OR
OR
60
ENCODER
ENABLE
2
1
DIR
COMP
8
8
WINDOW NUMBER
UPPER ADDRESS BITS

Fig. 21

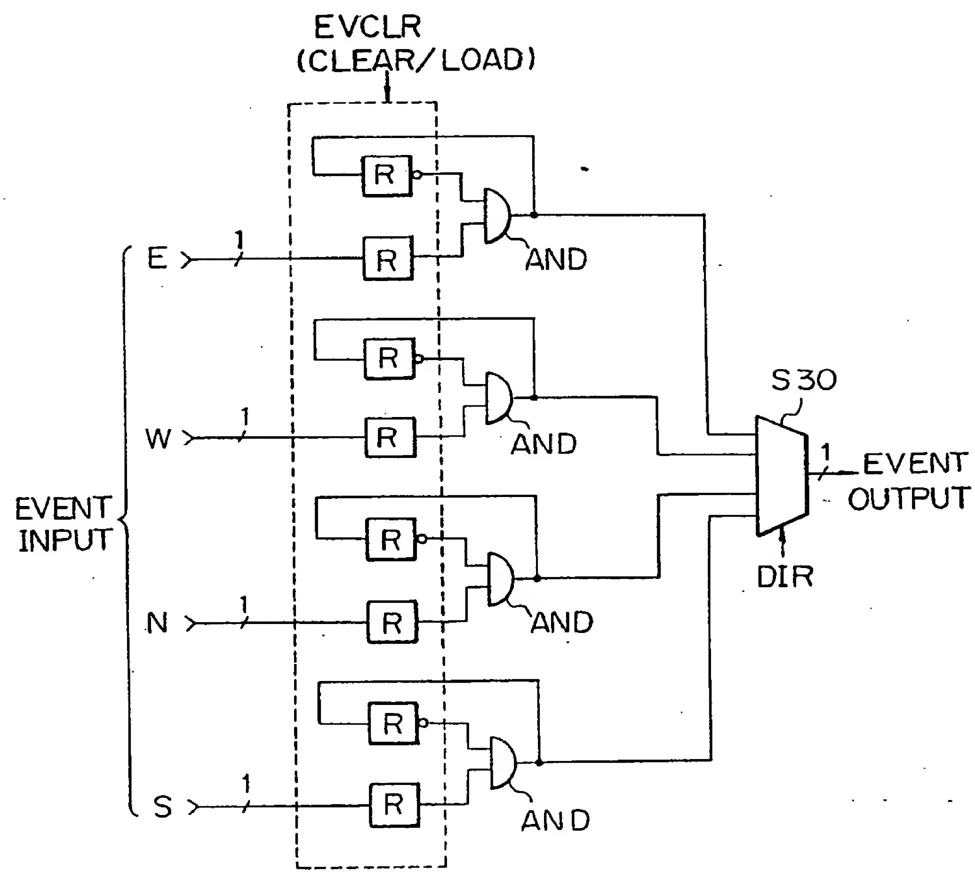

EVCLR
(CLEAR/LOAD)
R
R
AND
E
1
R
R
AND
W
1
EVENT
INPUT
R
R
AND
N
1
R
R
AND
S
1
S30
1
EVENT
OUTPUT
DIR

Fig. 22

INPUT				OUTPUT	
REGISTRATION OUTPUT	CONSECUTIVENESS OUTPUT	EVENT OUTPUT	SELF-EVENT	REGISTRATION	ADDRESS HOLDING
F	F	F	F	F	F
T	F	F	F	F	F
F	T	F	F	F	F
T	T	F	F		
F	F	T	F	T	F
T	F	T	F	F	F
F	T	T	F	F	T
T	T	T	F		
—	—	—	T	F	T

Fig. 23

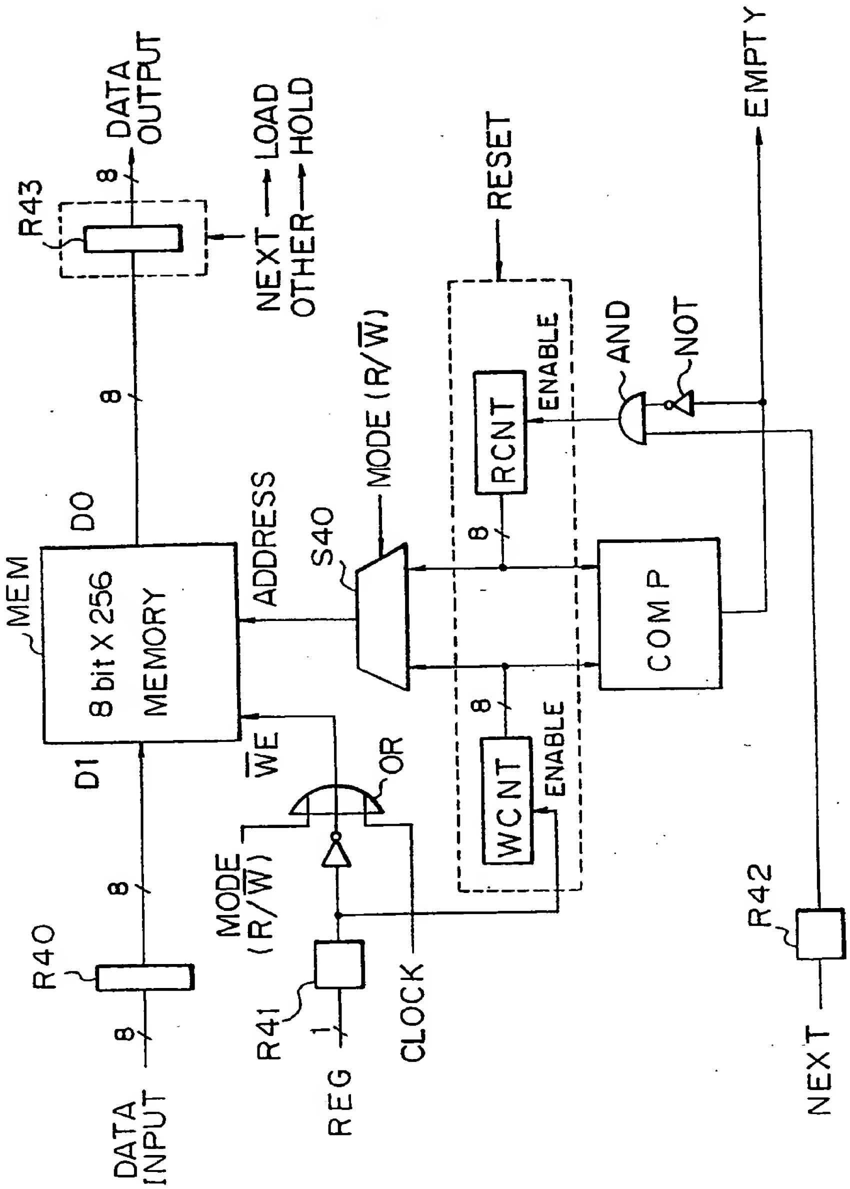

DATA INPUT
R40
8
8
D1
MEM
8 bit X 256 MEMORY
DO
8
8
R43
DATA OUTPUT
NEXT → LOAD
OTHER → HOLD
REG
1
R41
MODE (R/W̄)
CLOCK
OR
W̄E
ADDRESS
S40
MODE (R/W̄)
WCNT
ENABLE
8
RCNT
ENABLE
8
RESET
COMP
AND
NOT
EMPTY
NEXT
R42

Fig. 24

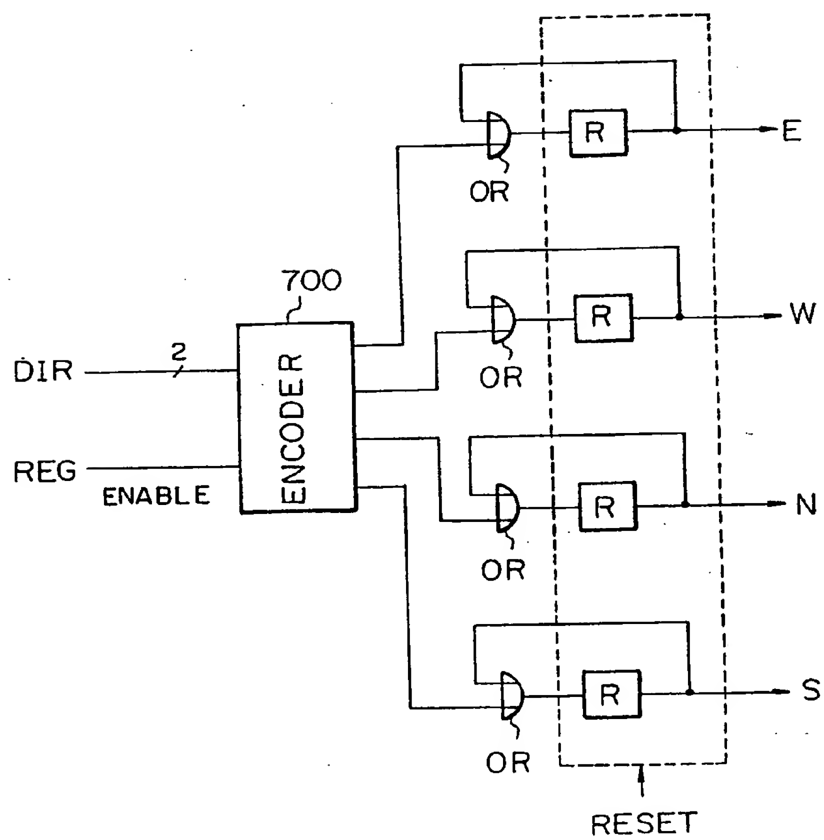


Fig. 25A

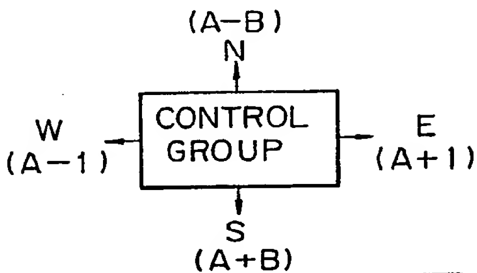


Fig. 25B

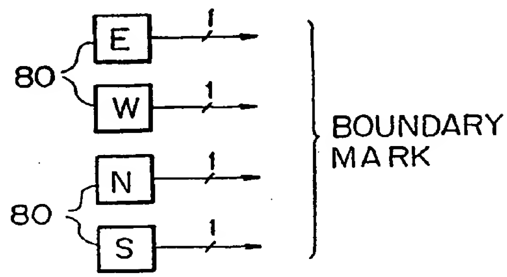


Fig. 25C

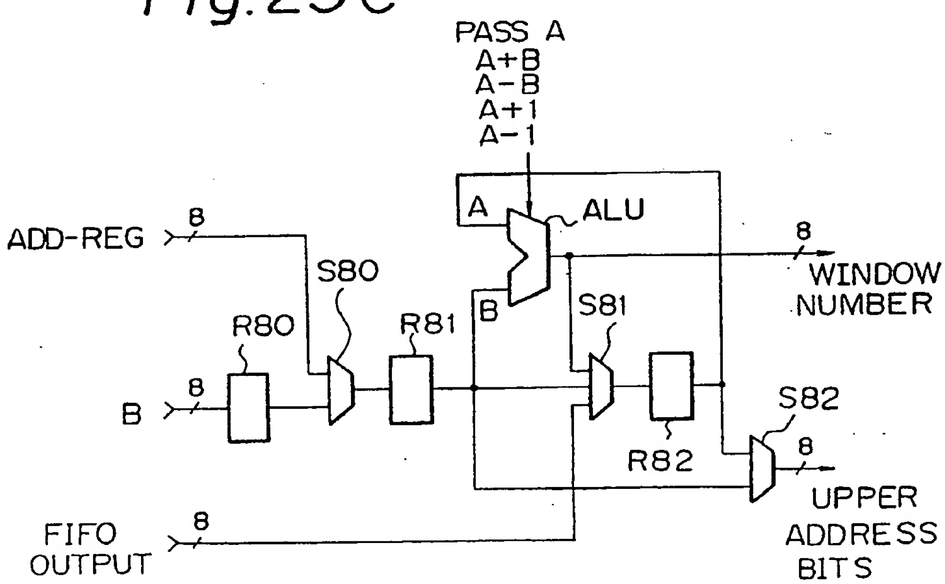
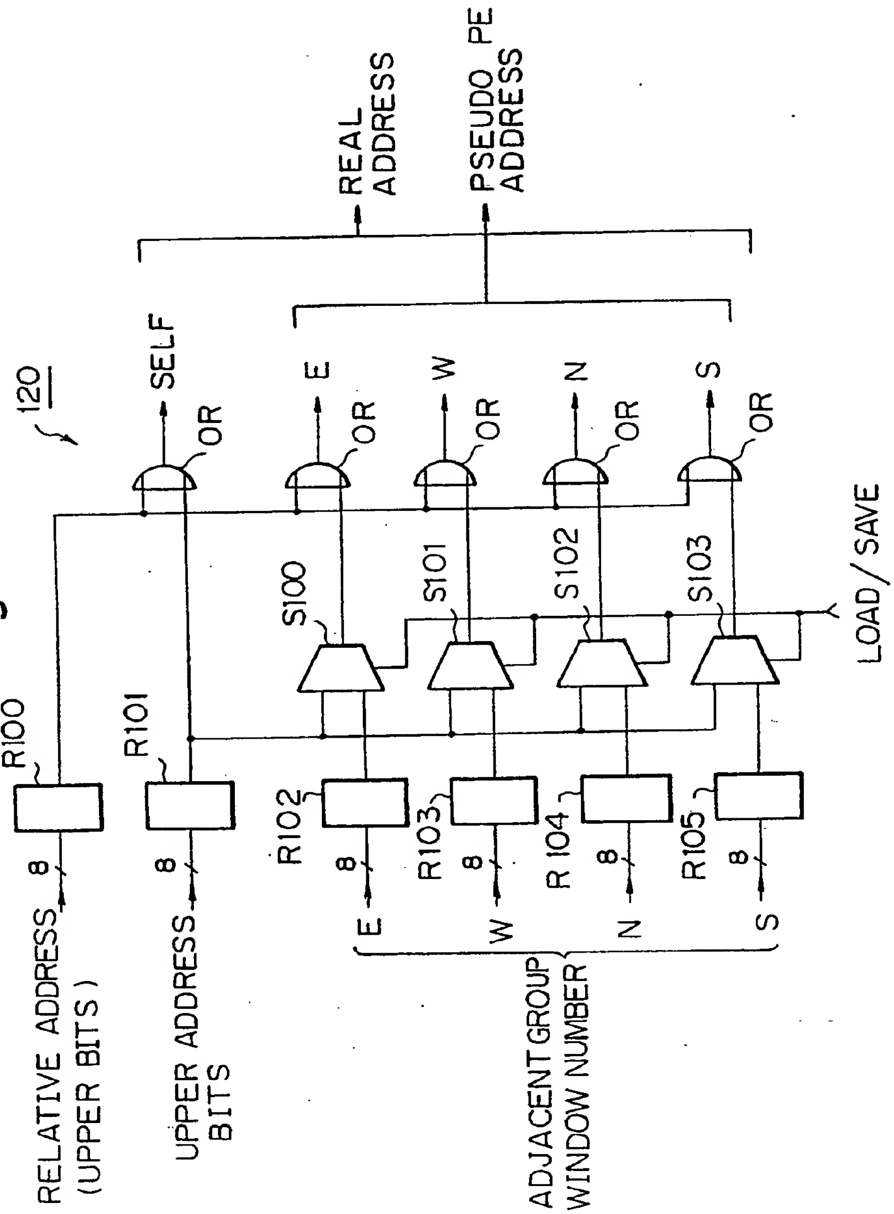


Fig. 26


120
RELATIVE ADDRESS (UPPER BITS)
UPPER ADDRESS BITS
ADJACENT GROUP WINDOW NUMBER
E
W
N
S
8
R100
R101
R102
R103
R 104
R105
S100
S101
S102
S103
LOAD / SAVE
OR
SELF
E
W
N
S
REAL ADDRESS
PSEUDO PE ADDRESS